DDC

LEVEL II

UNITED STATES AIR FORCE

AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

81 6 30 073

DISCLAIMER NOTICE

THIS DOCUMENT IS BEST QUALITY PRACTICABLE. THE COPY FURNISHED TO DTIC CONTAINED A SIGNIFICANT NUMBER OF PAGES WHICH DO NOT REPRODUCE LEGIBLY.

AFIT/GEO/EE/80D-12

ADAPTIVE LASER POINTING

AND

TRACKING PROBLEM

THESIS

AFIT/GEO/EE/80D-12 James Singletery Jr.
1st Lt USAF

Approved for public release; distribution unlimited

14

AFIT/GEO/EE/80D-12

6 ADAPTIVE LASER POINTING

AND

TRACKING PROBLEM

9 Master's THESIS

Presented to the Faculty of the School of Engineering

of the Air Force Institute of Technology

Air University

in Partial Fulfillment of the

Requirements for the Degree of

Master of Science in Electrical Engineering

12 112

by

10 James Singletery, Jr. B.S.E.E.

1st Lt USAF

Graduate Electro-Optics

11 Dece[redacted]80

A 23

Approved for public release; distribution unlimited.

012225

Preface

This report is the third in a series of research projects devoted to the development of an extended Kalman filter algorithm for use in a ground based laser system located at Kirtland AFB, New Mexico.

Much thanks goes to my thesis advisor, Dr. Maybeck, for his patience, consideration, and above all expert advice given throughout this project. Dr. Kabrinsky deserves additional thanks for his assistance in the pattern recognition area. Finally, without the help of my experienced typist, Ms. Cheryl Nicol, this report would not have been completed.

James Singletery Jr.

Contents

List of Figures

List of Figures

List of Tables

List of Symbols

List of Symbols

10 to the -8th power

Abstract

Although a number of the major objectives that were established at the outset of this project were not met, a number of milestones were realized. The digital implementation of a negating phase shift that operates perfectly under ideal conditions was a major accomplishment. The establishment of a zero level of 10^{-8} was also significant. The incorporation of the exponential smoothing technique to minimize the effect of measurement noise was important since it uncovered a possible connection between the size of the target image and its performance throughout the pattern recognition process. However, the major obstacle that surface during the execution of this project was a filter divergence problem. It has been proposed that this problem can be solved by implementing the Fourier transform derivative property instead of the forward-backward difference method to compute the spatial derivative of the non-linear h function.

I Introduction

Background

The application of laser technology to everyday life is growing in importance as each year passes. In areas from medicine to industry to military applications, laser technology, although in its embryonic stage, has gained a foothold and is destined to have a major impact on the future. For instance, in the industrial area, laser drilling has significantly improved the machining qualities resulting in smoother cuts, reduced tool force, and increased speed and accuracy (1:225-31). In the medical area, the application of lasers in opthalmology is promising. For example, the use of lasers to repair retinal tears and holes has been quite successful (2:360-7). For military applications, the ability to deposit large amounts of laser energy onto targets is a major research effort. One in a number of major obstacles before realizing this application deals with the precision tracking of a target and the subsequent pointing of the laser beam. This thesis is one in a series of research efforts devoted to solving this problem.

Traditionally, correlation algorithms have been employed to provide pointing and tracking information to a system. This correlation tracker stores a set of predetermined or previous real-time data target images in memory, compares these images with the images from its sensors. In the process of performing the mathematical calculations to characterize the differences between the stored and actual images, the appropriate commands are sent to the tracker to minimize the offset between the two (3:30-8). However, two major disadvantages of the correlation algorithm are its suspectibility to noise and absence of sensitivity to target

dynamics (3:31). To combat these drawbacks, the use of an extended Kalman Filter algorithm in place of the correlation tracker is being explored.

The Kalman Filter is a computer algorithm which processes noise-corrupted measurements and provides a reasonably accurate estimate of the state variables of interest (4:3). The actual mathematical details of the Kalman Filter are contained in Chapter 2 under the subtitle Extended Kalman Filter.

As mentioned earlier, this project is one in a series of research efforts. To be more precise, this project is the follow-on to two other projects. The first of these, entitled "An Extended Kalman Filter For Use in a Shared Aperture Medium Range Tracker" by Daniel E. Mercier, dealt with a very benign distant point target which could be analytically expressed as a two-dimensional gaussian intensity profile of circular contours (3:6-7). This intensity profile and all other profiles developed for this research effort are assumed to be scanned by a Forward-Looking Infrared (FLIR) sensor. This sensor horizontally and vertically scans the system field of view (FOV) to provide an 8 x 8 array of discrete values. These discrete values represent the average intensity of that portion of the image which lay across the appropriate detector (3:4-5).

Using some of the results of Mercier's Thesis, the second project, entitled "An Adaptive Distributed Measurement Extended Kalman Filter For a Short Range Tracker" by Robert L. Jensen and Douglas A. Harnly, dealt with more dynamic targets at closer ranges but still assumed that the target intensity pattern could be expressed analytically. However, Jensen and Harnly's thesis did make provisions to adaptively change the

shape of unimodal intensity patterns (5:77). Still, for actual hardware implementation, a priori knowledge of the analytic form of the intensity often cannot be assumed. Instead, on-line numerical techniques will have to be exploited to provide the necessary information to the Kalman Filter. The development of these techniques and their subsequent interface with the Kalman Filter is the basis for this project.

Problem Overview

The numerical techniques mentioned toward the end of the previous section include the Fast Fourier Transform (FFT), the shift theorem of Fourier Transform, the exponential smoothing technique, and the forward-backward difference method. These techniques are discussed in more detail in Chapter 2 and Appendix A. However, all involve the manipulation of intensity measurements provided by the FLIR sensor. The end result of these computations is to provide information to the Kalman Filter about the intensity pattern shape. This information takes the form of certain components of the measurement update equation. A detailed description of the measurement update equation is contained in Chapter 2 under the subtitle Measurement Update. However, the general form of this equation is

$$\underline{\hat{x}}(t_i^+) = \underline{\hat{x}}(t_i^-) + \underline{K}(t_i)\ (\underline{z}(t_i) - \underline{h}(\hat{x}(t_i^-)t_i) \tag{1}$$

where $\underline{\hat{x}}(t_i^+)$ = state estimate vector after incorporation of measurement

$\underline{\hat{x}}(t_i^-)$ = propagated state estimate vector before incorporation of measurement

$\underline{K}(t_i)$ = Kalman Filter Gain

$\underline{z}(t_i)$ = actual measurement vector

$\underline{h}(\hat{x}(t_i^-)t_i))$ = non-linear function of intensity measurement at time t_i, as a function of the true state estimate

In devising this project, it was decided that under ideal conditions, the Kalman Filter would provide state estimates which would center the target images from one sample period to another. This desirability of producing centered images is motivated by the simplicity it produces in the Kalman Filter equations. If the center images correspond to state estimates equal to zero, the resulting measurement equations become

$$\underline{\hat{x}}(t_i^-) = \underline{0} \tag{2}$$

$$\underline{\hat{x}}(t_i^+) = \underline{0} + \underline{K}(t_i)\ (z(t_i) - h(0_1 t_i)) \tag{3}$$

where

$\underline{h}(0_1 t_i)$ = centered non-linear h function

The generation of $\underline{h}(0_1 t_i)$ involves the use of the FFT, shifting theorem, and exponential smoothing techniques mentioned earlier. The details of how these techniques are utilized will be discussed later however, buried within the Kalman Filter gain ($\underline{K}(t_i)$) is the spatial derivative of the centered non-linear h function (see Chapter 2, Equation 56). This spatial derivative is generated using the forward-backward difference method discussed in Appendix A.

The flow of the data processing scheme is shown in Figure 1. In essence, there are two parallel data processing paths for the intensity measurements. The first path involves taking the 8 x 8 array of intensity measurements and arranging it by rows into a 64 x 1 measurement vector. This vector is then provided, as a measurement $\underline{z}(t_i)$, to the extended Kalman Filter which in turn provides state estimates that are used in the second path to provide centered measurement functions. This

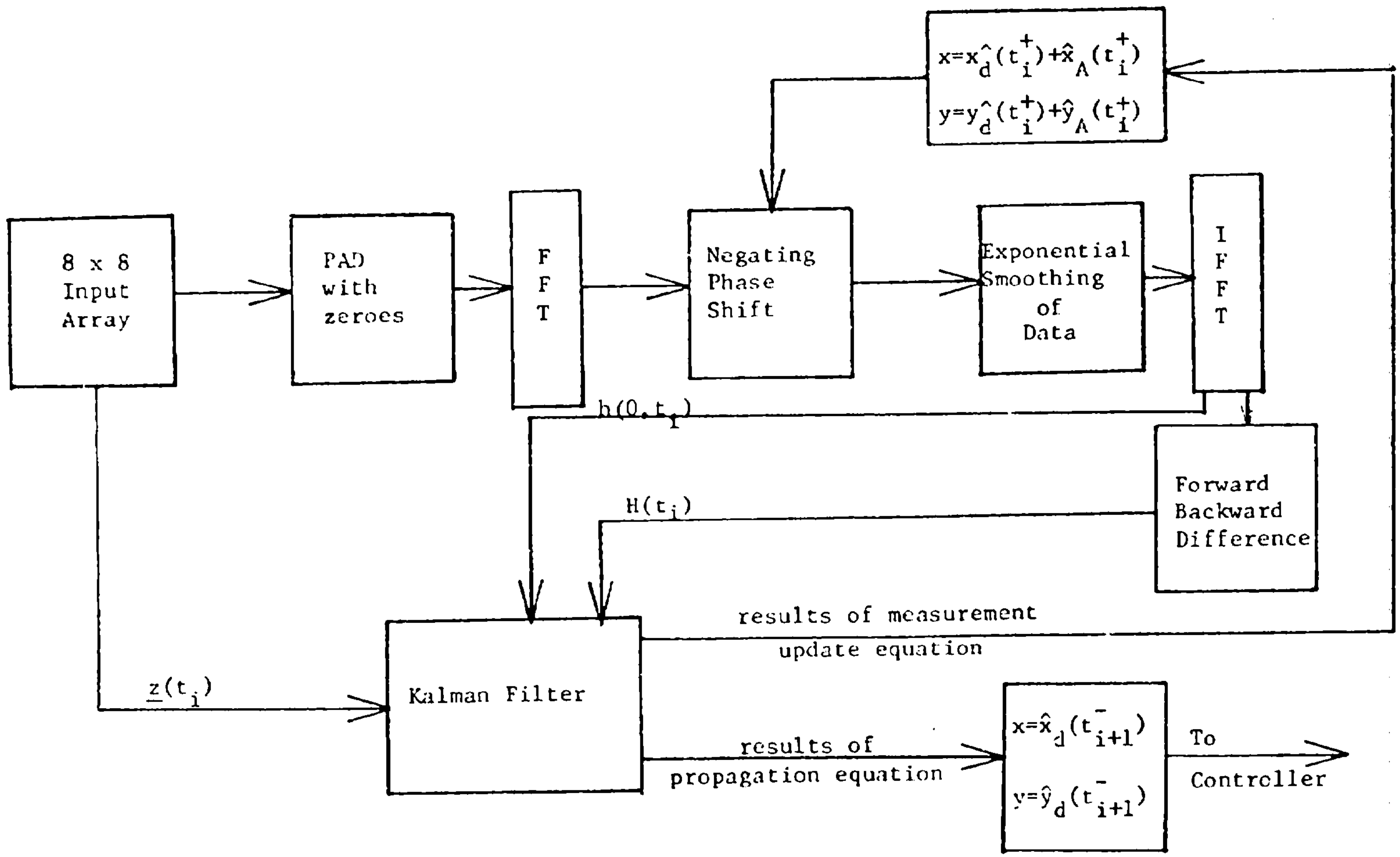

Figure 1. Data Processing Scheme

second path represents the main thrust of this thesis. Its purpose is to provide the centered non-linear and linear measurement functions mentioned earlier. To accomplish this, the shifting theorem of Fourier Transforms is exploited. In essence, the translation of an intensity pattern in the space domain can be negated by multiplying its Fourier Transform by the complex conjugate of the resulting linear phase shift (see Chapter 2, Fourier Transform). The source of the image translation about the FLIR array can be traced to two effects: the actual target dynamics and atmospheric jitter. An estimate of these effects are available from the Kalman Filter measurement update equations (see Chapter 2, Extended Kalman Filter). In turn, these best estimates are used in the argument of the complex conjugate of the linear phase shift to provide centered measurement functions. In addition, before the inverse Fourier Transform is taken, this centered pattern is averaged, using the exponential smoothing technique, with previous centered Fourier Transformed patterns to minimize the effect of measurement noise (see Chapter 2, Averaging). Once the inverse Fourier Transform is taken, the spatial derivative is then taken using the forward-backward difference approximation discussed in Appendix A to provide the centered linearized H function used in the calculation of the Kalman Filter Gain.

The sequential processing along the second path shown in Figure 1 together with its connection to the first path is an important fact. A copy of the 8 x 8 array of data is first padded with zeroes to alleviate problems in using the FFT (see Chapter 2, Fourier Transform). Next, the Fourier Transform of the two-dimensional array of data is taken. Before the negating phase shift is applied to this Fourier Transform, the processing of the measurement vector along the first path has to be

completed, since the resulting state estimates from the measurement update equation are used in the linear phase shift in the second data processing path. After applying the negating linear phase shift, this centered Fourier Transform is averaged with past centered Fourier Transforms to minimize the effect of noise. Next, the inverse Fourier Transform is taken which, theoretically, results in a centered pattern with noise effects substantially reduced. The zeroes that were initially added to the input array are stripped away with the remaining 8 x 8 data array representing the non-linear h function. In addition, using the numerical approximation discussed in Appendix A, the linearized H function is generated from this non-linear h function and both functions are used by the Kalman Filter for processing of the next measurement that becomes available.

An additional process that occurs along the first path is the propagation of the state estimate vector to the next sample time. This process provides the best prediction of where the target will be located just before the next measurement update is taken. Therefore, this information could be fed to a controller so as to minimize the perturbations of the image about the center of the FOV.

Plan of Attack

The plan of attack presented here provides a general flow of what will later be examined in greater detail in the performance analysis section. The verification of a pattern recognition algorithm along with its interface with the Extended Kalman Filter represents the basic premise upon which this thesis project was developed.

The verification of the pattern recognition algorithm is composed

of several different parts. These parts include verifying the FFT algorithm, verifying the ability to negate the translational effects in the space domain given perfect phase shift information to apply in the spatial frequency domain, and verifying the exponential smoothing technique as a means to minimize the effect of noise corruption on the measurement information.

The Extended Kalman Filter section involves measuring the impact of the non-linear and linearized h functions developed in the pattern recognition section on the performance of the Kalman Filter as compared to being given correct h functions. Also implicit in this verification is another comparison involving the use of state estimates from the Kalman Filter, instead of providing artificial knowledge of the true pattern offset, to provide the shift information needed in the pattern recognition section to provide centered patterns used to generate the h functions mentioned earlier (see Problem Overview).

Karhunen-Loëve Transformation

In the area of pattern recognition, it is highly desirable to determine the optimal set of eigenfunctions and their corresponding eigenvalues to represent two-dimensional intensity patterns. This representation can be determined with the orthogonal Karhunen-Loëve transformation

$$\int_{-y}^{y}\int_{-x}^{x} R(x,\alpha;y,\beta)\ \phi_i(\alpha,\beta)\ d\alpha d\beta = \lambda_i\ \phi_i(\alpha,\beta) \tag{4}$$

where $R(x,\alpha;y,\beta)$ = spatial autocorrelation kernel

$\phi_i(\alpha,\beta)$ = eigenfunctions

λ_i = eigenvalues

Based on properties specified through the spatial autocorrelation kernel, the entire information of an image is preserved by the given set of eigenvalues and eigenfunctions (6:6). Furthermore, since the low order terms normally provide the maximum sensitivity to target motion, the higher order terms of this spatial model decomposition can be neglected with little degradation in the quality of the resultant image (6:6).

As mentioned earlier, the key to finding the optimal set of functions is to know the spatial autocorrelation kernel. For intensity patterns represented by a two-dimensional discrete array of values, the correlation kernel is represented as a correlation matrix of dimension $N^2 \times N^2$ with N being the dimension of the square input matrix (6:6). The generation of this correlation matrix is a major drawback in using the Karhunen-Loëve transformation. However, if this matrix can be assumed or determined, the transformation matrix (A) which diagonalizes

the correlation matrix (R) can be evaluated by:

$$\Lambda^T R\Lambda = \begin{bmatrix} \lambda_1 & & \\ & \lambda_2 & \\ & & \ddots \lambda_{N^2} \end{bmatrix} \tag{5}$$

where λ_i's represents the eigenvalues in decreasing order of magnitude ($\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_{N^2}$) (6:6). Since the transformation matrix (A) is developed from orthogonal eigenvectors of the correlation matrix, the transformation matrix is orthogonal. Thus, the forward and reverse Karhunen-Loëve transformations that will preserve the quality of the image are

forward direction: $(F) = (f)\ (\Lambda)$

reverse direction: $(f) = (F)\ (\Lambda)^T$

where (f) = image vector

(F) = resultant image vector in the transform domain

(Λ) = transformation matrix (6:7)

Once again, if the entire group of eigenvalues and eigenvectors were retained, the total quality of the image would be preserved. But, if only m of the first N^2 eigenvalues were retained, the resultant mean square error (MSE) between the reconstructed image and the initial image would be the sum of the eigenvalues not included in the transformation.

$$MSE = \sum_{i=m+1}^{N^2} \lambda_i \quad (6:7)$$

As discussed in the article "Image Processing by Computer" by Guy Hanuise, from which the Karhunen-Loëve transformation argument has been developed, this transformation does possess some major disadvantages. The most significant drawbacks center around the generation of the correlation matrix. As the size of the square image matrix (N x N)

grows, the correlation matrix grows as N^2 (i.e. $N^2 \times N^2$). Thus, with large image quantization levels, serious data processing problems arise (6:7). Along the same lines, the exact calculation of the correlation matrix is very difficult to perform. As a result, more common orthogonal transformations such as the Fourier transform are used in signal processing. Implicit in the use of the Fourier transform are the assumptions of spatial stationarity of the autocorrelation kernel and a space domain infinite in extent. Referring back to equation 4, in applying the assumption of spatial stationary, and a domain of infinite extent, the Karhunen-Loève transformation equation becomes

$$\int_{-\infty}^{\infty} \int_{-\infty}^{\infty} R(x-\alpha, y-\beta)\phi_i(\alpha,\beta)\,d\alpha\,d\beta = \lambda_i \phi_i(\alpha,\beta) \tag{6}$$

Upon close examination of equation 6, the integral equation would be recognized as a two-dimensional convolution of two functions (7:10). The unique feature of the convolution theorem that makes it very appealing is that in the other transform domain, the two functions are multiplied together. Therefore, the Fourier transform of equation 6 can be written as

$$S(w_x, w_y)\ \Phi_i(jw_x, jw_y) = \lambda_i \Phi(jw_x, w_y) \tag{7}$$

where $S(w_x, w_y)$ = Fourier transform of $R(x,y)$ (power density spectrum)

$\Phi_i(jw_x, jw_y)$ = Fourier transform of $\phi_i(\alpha,\beta)$

For the equality in equation 7 to hold either one of two conditions must be met: either $S(w_x, w_y) = \lambda_i$, which is an impossibility since the λ_i's are scalar multiples and $S(w_x, w_y)$ is in a functional form, or more realistically, $\Phi_i(jw_x, jw_y)$ is an impulse function. The choice of the impulse function would be appropriate since it can be set to sample the

power density spectrum at the particular value of w_x's where the function equals the value of the scalar multiple.

$$\phi_i(jw_x, jw_y) = \delta(w_x - w_{xi}; w_y - w_{yi}) \tag{8}$$

Thus, by taking the inverse Fourier transform of this impulse function, the resulting eigenfunctions for the space domain are:

$$\phi_i(x,y) = \mathcal{F}^{-1}(\delta(w_x - w_{xi}; w_y - w_{yi})) \tag{9}$$

$$* \ \phi_i(x,y) = \exp(j(w_{xi}\lambda + w_{yi}\lambda)) \quad (8:237\text{-}42) \tag{10}$$

In reality, the space domain is limited by the system FOV and the random process describing the image intensity may not be truly spatially stationary. However, if the system FOV is relatively large and the random process is quasi-stationary, it could be heuristically argued that the resulting eigenfunctions still asymptotically approach complex exponentials. This result would provide reasonable motivation to use complex exponentials (Fourier transform) as the transformation function on the images in question.

In closing this section, it should be reiterated that the Karhunen-Loëve transformation is difficult to perform in its exact form. However, under the assumptions of spatial stationarity and a space domain large in extent, the Karhunen-Loëve equation provides adequate motivation for using familiar transformations involving complex exponentials such as Fourier.

* This same argument has been developed for the one-dimensional case in Chapter 8 of Introduction to Statistical Pattern Recognition by Keinosuke Fukunage.

Fourier Transform

The Fourier transform is a familiar transformation to the electrical engineer. In the one dimensional case, it is a transform quite often used to relate occurrences in the time domain to those in the frequency domain. However, with the ability of lenses to perform Fourier transforming instantaneously, the field of Fourier optics has provided motivation for extending the concept of the Fourier transform and its properties into two-dimensional space. As a result of this extension, the two-dimensional Fourier transform becomes

$$G(f_x,f_y) = \int_{-\infty}^{\infty}\int g(x,y)\ \exp -j2\pi(f_x x + f_y y)\ dxdy\ (7:5) \qquad (11)$$

where

$G(f_x,f_y)$ = Fourier or Frequency Spectrum

$g(x,y)$ = Function in the Space Domain

f_x,f_y = Spatial Frequencies

x,y = Spatial Variables

In comparing the one-dimensional and two-dimensional Fourier transforms, similarities should be recognizable; the use of the complex exponential as the eigenfunction, and the generation of spatial frequencies f_x,f_y to correspond with the spatial coordinate x and y. A particular property that has been extended to the two-dimensional case which is a vital part of this thesis is the shift theorem. The shift theorem states that a translation of an image in the space domain results in a linear phase shift in the spatial frequency domain:

$$T\{g(x-a,y-b)\} = G(f_x,f_y)\ \exp(-j2\pi(f_x a + f_y b))\ (7:9) \qquad (12)$$

where

a = offset of the spatial function along the x direction from a centered position

b = offset of the spatial function along the y direction from a centered position

To negate the translational effects in the space domain, the Fourier transform of the translated image is multiplied by the complex conjugate of the linear phase shift:

$$g(x,y) = T^{-1}\{G(f_x,f_y)\ \exp(-j2\pi(f_x a + f_y b))\exp(j2\pi(f_x a + f_y b))\} \qquad (13)$$

Up to now, the continuous case of the Fourier transform has been discussed, but to utilize the Fourier transform and its properties for computer simulation, the discrete form of the Fourier equations must be used. Such a development is contained in Digital Signal Processing by Alan V. Oppenheim and Ronald W. Schafer. The discrete Fourier transform and the discrete version of the shift theorem are as follows:

$$T(x(m,n)) = \frac{1}{MN}\sum_{k=0}^{M-1}\sum_{\ell=0}^{N-1} \chi(k,\ell)\exp\left(\frac{-j2\pi km}{M}\right)\ \exp\left(\frac{-j2\pi \ell n}{N}\right) \qquad (14)$$

Discrete Fourier Transform (9:115)

$$T(x(m-m_0,n-n_0)) = \frac{1}{MN}\sum_{k=0}^{M-1}\sum_{\ell=0}^{N-1} \chi(k,\ell)\exp\left(\frac{-j2\pi km}{M}\right)\ \exp\left(\frac{-j2\pi \ell n}{N}\right)$$

$$\times\ \exp\left(\frac{-j2\pi kmo}{M}\right)\ \exp\left(\frac{-j2\pi \ell no}{N}\right) \qquad (15)$$

$0 \leq m_o \leq M \qquad 0 \leq n_o \leq N$

Discrete Shift Theorem (9:110)

where m,n = sequence numbers in the space domain

k,ℓ = sequence numbers in spatial frequency domain

M,N = spatial area covered by the sequence numbers

m_o, n_o = translation of discrete pattern about its centered location.

As the definition of the variables may imply, the use of the discrete Fourier transform implies certain knowledge is known. In essence, the discrete Fourier transform views a finite area as being repeated indefinite in both the x and y directions. Therefore, the establishment of the two-dimensional periodicity is imperative. The discretzation of the finite area is reflected in the sequence numbers. The relationship between the sequence numbers and distance in the original space domain is shown in Figure 2. Notice that each pair of sequence numbers represent a smaller area within the previously mentioned finite area. For application to this thesis, the smaller area is represented by 20 urad x 20 urad with the total finite area being 480 urad x 480 urad. This results in the sequence numbers (m,n) varying from 1 to 24. Thus the sequence lengths (M,N) equal 24.

In the computer software for this project, a more efficient version of the discrete Fourier transform known as the Fast Fourier Transform (FFT) is used. The FFT and the Inverse Fast Fourier Transform (IFFT) are performed using a subroutine called FOURT. FOURT is a multidimensional FFT routine which uses the Cooley-Tukey method of calculation (10:76-9). A unique feature of the FOURT subroutine is the arrangement of the data array in the spatial frequency domain. As a result of the FOURT software, the FOURT format locates the D.C. term (zero frequency) in the upper left hand corner and the harmonics are misaligned as shown in Figure 3. In order to perform signal processing in the spatial frequency domain, the data array quadrants are switched, 2 with 4 and 1 with 3 (11:17). As shown in Figure 4, for an M by N dimension array, where M

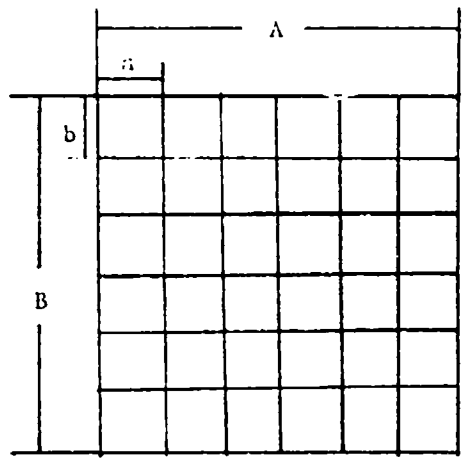

Figure 2A. Finite Area A x B

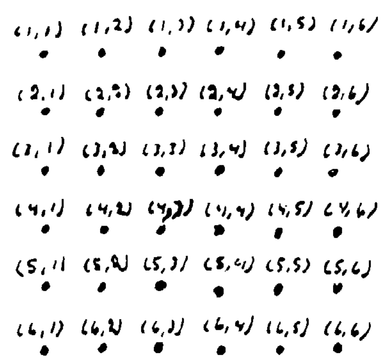

Figure 2B. Sequence Numbers

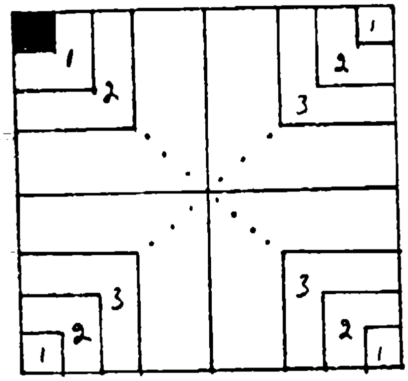

Figure 3. FOURT Format

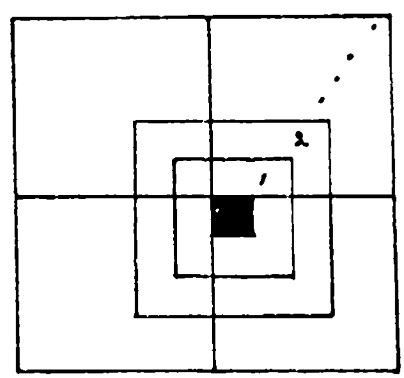

Figure 4. Rearranged Format

and N have to be even numbers, this rearranged format results in the D.C. term located at $(\frac{M}{2} + 1, \frac{N}{2} + 1)$ with the first concentric window surrounding the D.C. term being the first harmonic and corresponding higher harmonics matching with its appropriate window (11:18). Once the signal processing has taken place, the data array has to be re-arranged back into the FOURT format before the IFFT is taken (11:19-21).

Another requirement for FFT processing is to pad the input data array with zeroes to reduce edge effects, aliasing, and leakage conditions. A precise definition of all three of these conditions as they apply to FFT's is difficult to come by. However, the effect of these three conditions is related to how the FFT views the finite area as one period in a domain infinite in extent. For a finite area unpadded with zeroes, there is a chance that important information clustered along the edges can be viewed as part of the adjacent period thus causing distortion in the FFT and subsequent image from the IFFT (11:13, 92-3). To prevent this possible distortion from occurring with this project, the original 8 x 8 array of FLIR data is arbitrarily padded with an additional 8 rows of zeroes on the top, bottom, and sides. This additional padding results in the 24 x 24 input array mentioned earlier in this section.

In summary, the FFT and IFFT processes for this project is done using the FOURT subroutine. To negate the translational effects on an image, the shift theorem of the Fourier transform will be digitally implemented. Finally, to eliminate possible distortion by edge effects, aliasing and leakage conditions, the original 8 x 8 array of data is padded with additional zeroes.

Averaging

In reality, the existence of a noise-free environment is fictitious. For computer simulations, this noise could be added to a computer-generated noise-free image. The actual details of how the noise corruption simulation is accomplished is discussed in the next section. Nonetheless, the noise effects can be minimized by appropriate data processing techniques. Since a priori knowledge of the precise form of the noise is normally not available, the underlying mathematics makes little assumption about the precise form. However, it is necessary to assume that the noise changes faster from sample period to sample period than the image pattern itself. Traditionally, the moving average technique is used to combat the effect of noise under these conditions. This technique would store the most recent K-1 pieces of data in memory and average the data in memory with the new data in memory with the new piece of data using a weighting factor of 1/K on each piece of data (12:115). However, this technique does have one major disadvantage. It would require K storage locations of computer memory for each pixel. Therefore, for an N x N input array, KN^2 storage locations of computer memory is required. Thus, for large K and large input arrays, significant data processing problems would arise. An alternative method which alleviates this problem associated with the moving average is called exponential smoothing (12:114-28). The equation that is fundamental to this process is:

$$\hat{y}(t) = \alpha\, y(t) + (1-\alpha)\, \hat{y}(t-1) \quad (12:115) \tag{16}$$

where $\hat{y}(t)$ = most recent averaged value

$y(t)$ = most recent piece of data

$\hat{y}(t-1)$ = previous averaged value

α = smoothing constant

$0 \leq \alpha \leq 1$

A key parameter in the exponential smoothing equation is the smoothing constant α which can vary anywhere from 0 to 1 inclusive, depending on how much $\hat{y}(t)$ is to respond to the most recent piece of data. For noisy data and slowly changing signal pattern, it is suggested that the values of α tend more toward 0 than 1 so that there is some damping of the noise, yet there is still some response to the most recent piece of data (12: 115). For this thesis project, a steady state value of 0.1 for the smoothing constant was assumed. The notion of steady state value is important since a pseudo-exponential smoothing technique is used to provide better sensitivity to the initial 10 pieces of data. In essence, the smoothing constant is varied for each of the first ten runs (i.e. $\alpha = 1/K \quad K = 1,2,3...10$) until the steady state value of $\alpha = 0.1$ was reached.

In closing this section, it is interesting to note that the exponential smoothing technique possesses those qualities that the moving average lack. First, only $2N^2$ computer storage locations are needed to generate $\hat{y}(t)$. Second, $\hat{y}(t)$ contains portions of all past $\hat{y}(t)$'s although the initial pieces of data have less of an impact on the most recent averaged value.

Spatial Noise

To model the real world environment, noise corruption should be added to any computer simulation. There are two general types of characteristics to specify for the noise corruption in this problem, temporal and spatial. Temporally correlated noise implies that knowing

something about the noise at one particular time infers something about the noise at a later time. Normally, this inference can be modelled by a correlation function that is exponential in shape (5:37). However, since temporally correlated noise corruption has little effect on the quality of the image beyond that of temporally uncorrelated noise at the expected signal-to-noise ratio for this application, and since it is difficult to implement in a computer simulation, it was not included in this thesis project (5:37).

Under spatial noise characteristics, there are two categories, spatially uncorrelated and correlated noise. Before examining these two areas, a number system for the 8 x 8 input pixel array and the 64 x 64 resultant correlation matrix have to be developed. The 8 x 8 array is numbered from 1 to 64 by rows starting in the upper left-hand corner (Figure 5).

1	2	3	4	5	6	7	8
9	10	11	12	13	14	15	16
17	18	19	20	21	22	23	24
25	26	27	28	29	30	31	32
33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48
49	50	51	52	53	54	55	56
57	58	59	60	61	62	63	64

Figure 5. Pixel Numbering Scheme (5:19)

In the associated 64 x 64 correlation matrix, each row or column represents how that particular pixel relates to the other pixels in the input array. For example, row 1 or column 1 represents the correlation for how pixel 1 relates to itself and the other 63 pixels. Two unique

features of the correlation matrix are (1) since the correlation between two pixels is the same in both directions (i.e. $r_{ij} = r_{ji}$), the correlation matrix is symmetric and (2) since the correlation coefficient is one for a pixel related to itself, the diagonal elements of the matrix are one. In determining the off diagonal terms, the first and second nearest neighbor concept discussed in Harnly and Jensen's thesis was used (5:18-20). Essentially, the exponential form of the correlation function is assumed in generating the correlation coefficients. However, the non zero values generated are limited to the first and second nearest neighbors of the pixel in question. This condition results in 25 out of 64 entires in each row or column being nonzero. Of these 25 values, there are only six distinct values; one for the correlation coefficient of the pixel with itself, four values of 0.3679, four values of 0.2431, four values of 0.1353, eight values of 0.1069, and four values of 0.0591. These nonzero values, which are used to generate the 64 x 64 correlation coefficient matrix ($\underline{C}$), were developed from the following equation

$$C_{ij} = \exp\left(-\frac{d_{ij}}{20}\right) \tag{17}$$

where d_{ij} = distance from center of pixel i to center of pixel j

The choice of a correlation distance of 20 urad is dictated by the nonzero values generated by the first and second nearest neighbor concept (Figure 6).

$C_5 = 0.0591$	$C_4 = 0.1069$	$C_3 = 0.1353$	$C_4 = 0.1609$	$C_5 = 0.0591$
$C_4 = 0.1069$	$C_2 = 0.2431$	$C_1 = 0.3679$	$C_2 = 0.2431$	$C_4 = 0.1609$
$C_3 = 0.1353$	$C_1 = 0.3679$	$C_0 = 1$	$C_1 = 0.3679$	$C_3 = 0.1353$
$C_4 = 0.1069$	$C_2 = 0.2431$	$C_1 = 0.3679$	$C_2 = 0.2431$	$C_4 = 0.1609$
$C_5 = 0.0591$	$C_4 = 0.1069$	$C_3 = 0.1353$	$C_4 = 0.1609$	$C_5 = 0.0591$

Figure 6. First and Second Nearest Correlations Coefficient

For the spatially correlated case, the matrix of correlation coefficients becomes

$$\underline{C} = \begin{bmatrix} 1 & C_{1,2} & C_{1,3} & \cdots & C_{1,64} \\ C_{1,2} & 1 & C_{2,3} & \cdots & C_{2,64} \\ C_{1,3} & C_{2,3} & 1 & \cdots & C_{3,64} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ C_{1,64} & C_{2,64} & C_{3,64} & \cdots & 1 \end{bmatrix}$$

To complete the process for generating the noise, the 64 x 64 correlation

coefficient matrix is pre multiplied by the scalar value of the variance σ_n^2 in order to generate the correlation matrix itself.

$$\underline{R} = \sigma_n^2\,\underline{C} = \sigma_n^2 \begin{bmatrix} 1 & C_{1,2} & C_{1,3} & \cdots & C_{1,64} \\ C_{1,2} & 1 & C_{2,3} & \cdots & C_{2,64} \\ C_{1,3} & C_{2,3} & 1 & \cdots & C_{3,64} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ C_{1,64} & C_{2,64} & C_{3,64} & \cdots & 1 \end{bmatrix}$$

Now that $\underline{R}$ is known, the 64 x 1 noise vector $(\underline{V}(t_i))$, which will be added to the input array, can be computed by using the Cholesky square root. Specifically,

$$\underline{V}(t_i) = \mathcal{S}\underline{R}\;\underline{W}\;(t_i) \tag{18}$$

where $\underline{W}(t_i)$ = 64- dimensional independent white Gaussian noise vector (zero mean, variance of $\underline{I}$)

thus $E(\underline{W}(t_i)\ \underline{W}^T(t_i)) = \underline{I}\ \delta_{ij}$

The Cholesky square root is a unique matrix decomposition which produces the square root of $\underline{R}$ in the lower triangular form. This lower triangular form minimizes the number of nonzero compucations in generating $\underline{V}(t_i)$. To recover the original $\underline{R}$ matrix, the $\mathcal{S}\underline{R}$ should be post-multiplied by $\mathcal{S}\underline{R}^T$ therefore $\mathcal{S}\underline{R}\ \mathcal{S}\underline{R}^T = \underline{R}$. Because of this property, the covariance of $\underline{V}(t_i)$ is preserved with the use of the Cholesky square root:

$$E(\underline{V}(t_i)\ \underline{V}^T(t_j)) = E(\mathcal{S}\underline{R}\ \underline{W}(t_i)\ \underline{W}^T(t_j)\ \mathcal{S}\underline{R}^T)$$

$$= \mathcal{S}\underline{R}\ E(\underline{W}(t_i)\underline{W}^T(t_j))\mathcal{S}\underline{R}^T = \mathcal{S}\underline{R}\ \underline{I}\ \mathcal{S}\underline{R}^T\ \delta_{ij}$$

$$= \underline{R}\ \delta_{ij} \tag{19}$$

As mentioned earlier, the noise vector will be added to the noise-free input matrix to model real-world image conditions. This resulting vector called $\underline{z}(t_i)$ is the measurement vector that will be provided to the Extended Kalman Filter. Algebraically, the z vector is developed from the truth model and is of the form

$$\underline{z}(t_i) = \underline{h}(\underline{x}(t_i),\ t_i) + \underline{V}(t_i) \tag{20}$$

where $\underline{x}(t_i)$ = output state vector from the system dynamics model

$\underline{h}(\underline{x}(t_i),t_i)$ = nonlinear measurement function developed from image intensity pattern

$\underline{V}(t_i)$ = measurement noise vector

As discussed in the problem overview, the development of the nonlinear h function from the measurement vector $\underline{z}(t_i)$ is a major goal of this thesis.

In summary, since temporally correlated noise has been shown to have little effect on the Kalman Filter performance at the signal-to-noise ratios for this project, only spatially correlated and uncorrelated noise was used. The correlation matrix used to generate the noise vector $\underline{V}(t_i)$ is developed from the first and second nearest neighbor concept as it applies to a correlation function exponential in nature. The Cholesky square root of the correlation matrix is taken and post-multiplied by a white noise vector $\underline{W}(t_i)$, with the final result being the noise vector $\underline{V}(t_i)$ that is added to the noise-free image array to provide the necessary noise corruption.

Relationship Between Truth Model and Extended Kalman Filter

In the previous section, terms such as truth model, system dynamics, and Extended Kalman Filter were used. These terms represent the heart

of this project. The truth model is a best representation of the environment from which the measurement vector $\underline{z}(t_i)$ is generated. These environmental characteristics include the underlying target dynamics, atmospheric effects, and background noise. In order to process the measurement vector $\underline{z}(t_i)$ and provide a best estimate of the underlying target centroid location for the next sample period, the Extended Kalman Filter is used. The actual details of the filter will be discussed in a later section. However, the interplay between the Kalman Filter and the truth model is a point which at times can be confusing. In order to provide a best estimate of the underlying centroid location, the dynamic models are a facsimile of those in the truth model. Ideally, the models should match exactly, but in reality, due to the desire for computational efficiency, there are always deviations. The amount of deviation in the dynamics models for the filter varies depending on the problem application. A contributing factor to this deviation may be a lack of knowledge concerning the exact form of the truth models. However, due to the robustness of the Kalman filter, adequate predictions of the centroid location can still be accomplished.

Truth Model

Much has been discussed concerning the dynamic models. For this project, it had been envisioned to investigate the feasibility of using both deterministic and stochastic models. In the realm of deterministic models, initially, the target dynamics involved a stationary target centered in the system FOV. This step was taken to identify and eliminate any possible inherent motion caused by the pattern recognition or Extended Kalman Filter algorithms. The next deterministic model involved

a target moving acorss the image plane at a constant velocity. This was done to test the robustness of the Extended Kalman Filter. Supposedly, the filter design used for this project can absorb any truth model whose resultant motion from one sample period to the next is less than one-half pixel (i.e. less than 10 urads) (13).

In the realm of stochastic models, the target and atmospheric dynamics developed in Mercier's thesis were used (3:9-16). The target dynamics are modelled in each direction as a first-order Gauss-Markov process driven by white Gaussian noise

$$\dot{X}_D(t) = -\frac{1}{\lambda_T} X_D(t) + W_1(t) \tag{16}$$

$$\dot{Y}_D(t) = -\frac{1}{\lambda_T} Y_D(t) + W_2(t) \tag{17}$$

where $E(W_1(t)) = E(W_2(t)) = 0$ (18)

$$E(W_1(t)\ W_2(s)) = E(W_2(t)\ W_2(s)) = \frac{2\sigma_d^{\ 2}}{\lambda_T}\ \delta(t-s) \tag{19}$$

$$E(W_1(t)\ W_2(s)) = 0 \tag{20}$$

λ_T = truth model correlation time

$W_1(t)$, $W_2(t)$ = continuous time independent white Gaussian noise processes

$\sigma_d^{\ 2}$ = the desired variance on the outputs X_D and Y_D (3:10).

The atmospheric jitter model was based on a study by the Analytic Sciences Corporation (14:29,30) and a data analysis by Hogge and Butts (15). These studies resulted in the development of a third order shaping filter, with a single pole at 14.14 rad/sec and a double pole at 659.5 rad/sec, that is driven by white Gaussian noise (3:12).

Since any n^{th} order differential equation can be written as a set

of coupled first order differential equations, the state space model incorporating both the target dynamics and atmospheric effects was generated using the Jordon canonical form (3:13). Referring to the state space model from Mercier's thesis,

$$\dot{\underline{X}}_T = \underline{F}_T \underline{X}_T + \underline{G}_T \underline{W}_T (t) \tag{21}$$

where $\underline{F}_T$ = the truth model plant matrix

$\underline{X}_T$ = the truth model state vector

$\underline{G}_T$ = the truth model input matrix

$\underline{W}_T$ = a vector of white Gaussian noise inputs

$$E\{\underline{W}_T(t)\} = 0 \tag{22}$$

$$E\{W_T(t)\ W_T^T(s)\} = \underline{Q}_T\ \delta(t-s) \tag{23}$$

$$\underline{Q}_T = \begin{bmatrix} \frac{2\sigma_d^2}{\lambda_T} & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{2\sigma_d^2}{\lambda_T} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{24}$$

$$\underline{F}_T = \begin{bmatrix} -\frac{1}{\lambda_T} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -a & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -b & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -b & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -\frac{1}{\lambda_T} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -a & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -b & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -b \end{bmatrix} \tag{25}$$

$a = 14.14$ rad/sec $\quad b = 659.5$ rad/sec

$$\underline{G}_T = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & G_1 & 0 & 0 \\ 0 & G_2 & 0 & 0 \\ 0 & G_3 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & G_1 \\ 0 & 0 & 0 & G_2 \\ 0 & 0 & 0 & G_3 \end{bmatrix} \tag{26}$$

$$G_1 = \frac{ab^2}{(a-b)^2} \quad G_2 = -G_1 \quad G_3 = \frac{ab^2}{(a-b)} \qquad (3{:}11\text{-}14)$$

The composition of these truth model matrices is dictated by the components of the truth model state vector describing the target dynamics and atmospheric effect on the centroid location. For this project from $\underline{X}_T$,

$$X_D = X(1) \tag{27}$$

$$X_A + X(2) + X(3) \tag{28}$$

$$Y_D = X(5) \tag{29}$$

$$Y_A = X(6) + X(7) \quad (3{:}14) \tag{30}$$

where X_A and Y_A are atmospheric jitter variables as output of third order shaping filter.

In order to transition the state vector from one sample period to another, propagation equations are developed. The basis for these equations is the solution to the matrix differential equation

$$\dot{\underline{\Phi}}(t,t_i) = \underline{F}_T\, \underline{\Phi}(t,t_i) \tag{31}$$

where $\underline{\Phi}(t,t_i)$ = state transition matrix from time t_i to time t associated with the matrix $\underline{F}_T$

$$\underline{\Phi}(t_i,t_i) = \underline{I} \quad (3{:}14) \tag{32}$$

Since the truth model plant matrix $\underline{F}_T$ is a constant, the state transition matrix is stationary with respect to time. Therefore, $\underline{\Phi}(t_{ix},t_i)$ is also constant for a fixed sample period (3:14). Using this truth model plant matrix, the solution to the above differential equation becomes

$$\underline{\Phi}_T(\Delta t) = \begin{bmatrix} e^{-\Delta t/\lambda T} & 0 & 0 & 0 \\ 0 & e^{-a\Delta t} & 0 & 0 \\ 0 & 0 & e^{-b\Delta t} & te^{-b\Delta t} \\ 0 & 0 & 0 & e^{-b\Delta t} \end{bmatrix} \tag{33}$$

$$\Delta t = (t_{i+1} - t_i)$$

The state transition matrix is used in the solution to the state space vector differential equation

$$\dot{\underline{X}}_T = \underline{F}_T \ \underline{X}_T(t) + \underline{G}_T \ \underline{W}_T(t) \tag{34}$$

This solution is

$$\underline{X}_T(t_{i+1}) = \underline{\Phi}_T(\Delta t) \ \underline{X}_T(t_i) \tag{35}$$

$$+ \int_{t_i}^{t_{i+1}} \underline{\Phi}_T(t_{i+1},\lambda) \ \underline{G}_T(\lambda) \ \underline{W}_T(\lambda)d\lambda \tag{36}$$

The details for the following argument are contained in Mercier's thesis. Nonetheless, the resulting equivalent discrete-time model to the above equation is

$$\underline{X}_T(t_{i+1}) = \underline{\Phi}_T(\Delta t) \ \underline{X}_T(t_i) + \sqrt{Q}_d \ \underline{W}_d(t_i) \tag{37}$$

where $\sqrt{Q}_d$ is the Cholesky square root of

$$Q_D = \int_{t_i}^{t_{i+1}} \underline{\Phi}_T(t_{i+1},\lambda) \ \underline{G}_T(\lambda) \ \underline{Q}_T(\lambda) \ \underline{G}_T(\lambda) \ \underline{\Phi}_T^{T}(t_{i+1},\lambda)d\lambda \tag{38}$$

$$E\{\underline{W}_d(t_i)\} = \underline{0} \tag{39}$$

$$E\{\underline{W}_d(t_i) \ \underline{W}_d^{T}(t_i)\} = \underline{I} \ \lambda_{ij} \quad (3:14\text{-}5) \tag{40}$$

In summary, three truth models were developed for this project. Two models were deterministic in nature; a stationary target and a target moving at a constant velocity across the image plane. These models were primarily developed to trouble-shoot and provide a bench mark for the pattern recognition and Extended Kalman Filter algorithms. The third model is the stochostic representation that will ultimately be used to analyze the Filter performance.

Extended Kalman Filter

As stated before, the basic Extended Kalman Filter developed in Mercier's thesis was used for this project. The target dynamics assumed

by the Filter is also a first order Gauss-Markov process, generated by a first-order lag driven by white Gaussian noise (3:19). The differential equation describing a particular state, which is very similar to that for the stochastic truth model, is

$$\dot{X}(t) = -\frac{1}{\lambda} X(t) + W(t) \tag{41}$$

$$E(W(t)) = 0 \tag{42}$$

$$E(W(t)\ W(s)) = \frac{2\sigma^2}{\lambda}\ \delta(t-s) \tag{43}$$

where λ = correlation time

$X(t)$ = state

$\dot{X}(t)$ = time derivative of the state

$W(t)$ = white Gaussian noise (3:19)

However, the plant, input, and white noise covariance matrices assumed by the filter differ from those developed in the truth model. The reduced order filter design assumes four states in its state vector $(X_D, Y_D, X_A, Y_D)^T$ which results in a state vector differential equation

$$\underline{\dot{X}}_F(t) = \underline{F}_F\ \underline{X}_F(t) + \underline{W}_F(t) \tag{44}$$

where $\underline{X}_F(t)$ = filter state vector

$\underline{F}_F$ = filter plant matrix

$\underline{W}_F(t)$ = input white noise vector

$$\underline{F}_F = \begin{bmatrix} -\frac{1}{\lambda_D} & 0 & 0 & 0 \\ 0 & -\frac{1}{\lambda_D} & 0 & 0 \\ 0 & 0 & -\frac{1}{\lambda_A} & 0 \\ 0 & 0 & 0 & -\frac{1}{\lambda_A} \end{bmatrix} \tag{45}$$

λ_D = correlation time assumed for target dynamics

λ_A = correlation time assumed for atmospheric jitter

$$E(\underline{W}_F(t)) = 0 \tag{46}$$

$$E(\underline{W}_F(t)\ \underline{W}_F^T(s)) = \underline{Q}_F\ \delta(t-s) \tag{47}$$

$$\underline{Q}_F = \begin{bmatrix} \frac{2\sigma_D^2}{\lambda_D} & 0 & 0 & 0 \\ 0 & \frac{2\sigma_D^2}{\lambda_D} & 0 & 0 \\ 0 & 0 & \frac{2\sigma_A^2}{\lambda_A} & 0 \\ 0 & 0 & 0 & \frac{2\sigma_A^2}{\lambda_A} \end{bmatrix} \tag{48}$$

σ_D^2 = assumed target dynamics noise variance

σ_A^2 = assumed atmospheric noise variance (3:20)

It should be mentioned that the values of the $\underline{Q}_F$ matrix are determined during the off-line tuning process designed to produce the optimal tracking performance (16:224).

As with the truth model, the Kalman Filter performs a propagation of its state estimate vector and conditional covariance matrix from one sample time to the next. The state transition matrix is also developed from a differential equation similar to that used in the truth model, except the filter plant matrix is used instead of the truth model plant matrix.

$$\dot{\underline{\Phi}}_F(t,t_i) = \underline{F}_F\ \underline{\Phi}_F(t,t_i) \quad (3:21) \tag{49}$$

$$\underline{\Phi}(t_i, t_i) = \underline{I} \tag{50}$$

Since $\underline{F}_F$ is a time invariant matrix, the solution to the above differential equation is $\underline{\Phi}_F(t, t_i)$, the filter state transition matrix, given by

$$\underline{\Phi}_F(t, t_i) = \begin{bmatrix} e^{-(t-t_i)/\lambda_D} & 0 & 0 & 0 \\ 0 & e^{-(t-t_i)/\lambda_D} & 0 & 0 \\ 0 & 0 & e^{-(t-t_i)/\lambda_A} & 0 \\ 0 & 0 & 0 & e^{-(t-t_i)/\lambda_A} \end{bmatrix} \tag{51}$$

In solving the associated differential equation for the propagation of the covariance matrix, the following stochastic integral equation results:

$$\underline{P}(t_{i+1}^-) = \underline{\Phi}_F(t_{i+1}, t_i)\, \underline{P}(t_i^+)\, \underline{\Phi}_F^T(t_{i+1}, t_i)$$

$$+ \int_{t_i}^{t_{i+1}} \underline{\Phi}_F(t_{i+1}, \lambda)\, \underline{Q}_F(\lambda)\, \underline{\Phi}_F^T(t_{i+1}, \lambda)\, d\lambda \tag{52}$$

where $\underline{\Phi}_F(t_{i+1}, t_i)$ = filter state transition matrix

$\underline{P}(t_i^+)$ = conditional state covariance matrix from measurement update equation at time t_i

$\underline{P}(t_{i+1}^-)$ = conditional state covariance matrix propagated from time t_i to t_{i+1}

$\underline{Q}_F(\lambda)$ = noise covariance matrix (3:21)

Using the $\underline{Q}_F$ matrix specified in equation 48, the solution to the integral term in the stochastic equation above becomes

$$\begin{bmatrix} \sigma_D^2(1-e(-\frac{2\Delta t}{\lambda_D})) & 0 & 0 & 0 \\ 0 & \sigma_D^2(1-e(-\frac{2\Delta t}{\lambda_D})) & 0 & 0 \\ 0 & 0 & \sigma_A^2(1-e(-\frac{2\Delta t}{\lambda_A})) & 0 \\ 0 & 0 & 0 & \sigma_A^2(1-e(-\frac{2\Delta t}{\lambda_A})) \end{bmatrix} \tag{53}$$

To solidify the development of the propagation equations, the following summary of the estimated state vector and the conditional covariance matrix propagation equations is presented.

$$\underline{\hat{X}}(t_{i+1}^-) = \begin{bmatrix} e^{-\Delta t/\lambda_D} & 0 & 0 & 0 \\ 0 & e^{-\Delta t/\lambda_D} & 0 & 0 \\ 0 & 0 & e^{-\Delta t/\lambda_A} & 0 \\ 0 & 0 & 0 & e^{-\Delta t/\lambda_A} \end{bmatrix} \underline{\hat{X}}(t_i^+) \tag{54}$$

$$\underline{P}(t_{i+1}^-) = \begin{bmatrix} e^{-\Delta t/\lambda_D} & 0 & 0 & 0 \\ 0 & e^{-\Delta t/\lambda_D} & 0 & 0 \\ 0 & 0 & e^{-\Delta t/\lambda_A} & 0 \\ 0 & 0 & 0 & e^{-\Delta t/\lambda_A} \end{bmatrix} P(t_i^+)$$

$$\cdot \begin{bmatrix} e^{-\Delta t/\lambda_D} & 0 & 0 & 0 \\ 0 & e^{-\Delta t/\lambda_D} & 0 & 0 \\ 0 & 0 & e^{-\Delta t/\lambda_A} & 0 \\ 0 & 0 & 0 & e^{-\Delta t/\lambda_A} \end{bmatrix}$$

$$
+ \begin{bmatrix} \sigma_D^2(1-e(-\frac{2\Delta t}{\lambda_D})) & 0 & 0 & 0 \\ 0 & \sigma_D^2(1-e(-\frac{2\Delta t}{\lambda_D})) & 0 & 0 \\ 0 & 0 & \sigma_A^2(1-e(-\frac{2\Delta t}{\lambda_A})) & 0 \\ 0 & 0 & 0 & \sigma_A^2(1-e(-\frac{2\Delta t}{\lambda_A})) \end{bmatrix} \tag{55}
$$

Measurement Update

Once the state vector and covariance matrix are propagated up to the next sample time, the Kalman filter utilizes this information, along with the measurement vector available, to compute an estimate of the underlying target centroid location. The measurement update equations which are exploited by the Kalman Filter to perform this operation are:

$$
\underline{K}(t_i) = \underline{P}(t_i^-)\ \underline{H}^T(t_i)\ (\underline{H}(t_i)\underline{P}(t_i^-)\underline{H}(t_i) + \underline{R}(t_i))^{-1} \tag{56}
$$

$$
\hat{\underline{x}}(t_i^+) = \hat{\underline{x}}(t_i^-) + \underline{K}(t_i)\ (\underline{z}(t_i) - \underline{h}(\hat{x}(t_i^-),\ t_i)) \tag{57}
$$

$$
\underline{P}(t_i^+) = \underline{P}(t_i^-) - \underline{K}(t_i)\underline{H}(t_i)\underline{P}(t_i^-) \tag{58}
$$

where

$\underline{P}(t_i^-)$ = propagated conditional covariance matrix from filter, before measurement incorporation at time t_i

$\hat{\underline{x}}(t_i^-)$ = propagated state estimate vector from filter

$\underline{K}(t_i)$ = Kalman Filter Gain

$\underline{h}(\hat{x}(t_i^-),\ t_i)$ = non-linear function of intensity measurements at time t_i, as a function of the state estimate

$\underline{H}(t_i) = \dfrac{\partial \underline{h}(\hat{x}(t_i^-),\ t_i)}{\partial \underline{x}}$ = linear function of intensity measurements

$\underline{z}(t_i)$ = actual measurement vector

To ease the computational burden on the computer, a different form of the measurement update equation called the inverse covariance form is actually used in the computer software (3:26-7). In the conventional form, the calculation of the Kalman Filter Gain requires the inversion of a 64 x 64 matrix for each update, since there are 64 scalar measurements (3:26). However, in the inverse covariance form, only two 4 x 4 inverses are performed to obtain the $\underline{P}(t_i^-)$ and $\underline{P}(t_i^+)$ matrices (3:26-7). The equations for the inverse covariance form are as follows:

$$\underline{P}^{-1}(t_i^+) = \underline{P}^{-1}(t_i^-) + \underline{H}^T(t_i)\underline{R}(t_i)\underline{H}(t_i) \tag{59}$$

$$\underline{P}(t_i^+) = (\underline{P}^{-1}(t_i^+))^{-1} \tag{60}$$

$$\underline{K}(t_i) = \underline{P}(t_i^+)\underline{H}^T(t_i)\underline{R}^{-1}(t_i) \tag{61}$$

$$\underline{\hat{x}}(t_i^+) = \underline{\hat{x}}(t_i^-) + \underline{K}(t_i)\ (\underline{z}(t_i) - \underline{h}(\hat{x}(t_i),\ t_i)) \tag{62}$$

Normally, the non-linear h function and its spatial derivative (linearized H function) which are used in the update equations are explicit in form. However, for this project the non-linear h function is determined in real-time using the FFT, phase shifting and the averaging techniques discussed earlier. While the linearized H function is determined from the non-linear h function using the numerical approximation discussed in Appendix A.

In summary, the measurement update equation is that portion of the Kalman Filter which incorporates the actual intensity measurements to provide a best estimate of the target centroid location. In order to ease the computational burden of inverting a 64 x 64 matrix, the inverse

covariance form of the update equations, which only requires the inversion of two 4 x 4 matrices, was employed.

III Performance Analysis

The performance analysis for this thesis follows that outlined in the Plan of Attack section. To begin, the validation of the FFT subroutine FOURT not only demonstrated its ability to reconstruct a two-dimensional array of data, but also established a zero level of 10^{-8}. This zero level represents the greatest non-zero value in the IFFT where a zero would be found in the original image array.

The next step involved the development and validation of a subroutine used to perform the negating phase shift. This subroutine called Shift is described in more detail in Appendix D. In short, this subroutine applies the complex conjugate of the resulting linear phase shift from the translation of an image in the space domain to the Fourier transform of this same image. As a result, when the IFFT is taken, an image which is centered in the FOV is the end product. Figure 7 illustrates this process using the 40 urad x 40 urad intensity pattern.

In analyzing the performance of the exponential smoothing technique, a number of parameters were varied using the 3 gaussian pattern as a basis. These parameters included the signal-to-noise ratio, which varied between 10 and 20, and the sigma values for the spread of the gaussian patterns which included 1/4 pixel, 1 pixel, and 3 pixel spreads. The various sigma values provided a spectrum of shapes which ranged from sharply peaked (σ = 1/4 pixel) to very broad (σ = 3 pixels). The actual results are displayed in Figures 8 to 13. However, it is important to note that at a signal-to-noise ratio of 20, the pattern is readily identifiable irregardless of its shape. However, the average technique does provide some smoothing of the data. At a signal-to-noise ratio of 10, the shape of the

pattern seems to make a difference. For the narrow gaussian patterns (σ = 1/4 pixel) the pattern is difficult to recognize even after 50 runs. But, for the very wide gaussian pattern the general trend of increasing and decreasing numbers is maintained throughout the process, which may suggest that the filter performance may be better for larger targets. Still, it should be emphasized that this is only a superficial analysis and the more detailed analysis can only be performed once a complete filter simulation has been developed.

In implementing the first truth model, that of a stationary target, a major problem arose. Under ideal conditions (i.e. the Kalman Filter knowing exactly where the target is initially), the filter began to diverge from the known target. This divergence initially appeared at the first measurement update and became progressively worse as each subsequent propagation and measurement update were performed. Initially, it was speculated that there may be a Kalman Filter tuning problem. As a result, various values for the initial covariance matrix ($\underline{P}_0$) to improve the initial response of the filter, and for the covariance matrices for the dynamic driving noise ($\underline{Q}$) and the measurement noise ($\underline{R}$) to improve the steady state response were tried. Nonetheless, no noticeable improvement in the filter's performance resulted from this investigation. Next, it was conjectured that the problem may be within the subroutine UPD which performs the Kalman Filter measurement update. First, the search for a possible sign error or any other fault in the software was performed without any satisfaction. However, upon investigating the calculation of the Kalman filter gain $\underline{K}(t_i)$, it was hypothesized that the forward-backwards difference method used to calculate the linearized h function ($\underline{H}(t_i)$) which in turn is used to calculate the

filter gain, may be the source of the problem. To eliminate any possible concern that the divergence problem may be caused by the generation of the unknown non-linear and linear h functions, the analytic function for the noise-free centered gaussian patterns generated from the subroutine Input 3 was used as the non-linear h function. The analytic spatial derivative of the gaussian pattern in both the x and y directions was next taken and used as the linearized h function. The results of this analysis is presented in Table 1. As can be seen, it is verified that the source of the problem is not due to the generation of the unknown h functions. As a possible solution to this dilemma involving the divergence of the Kalman Filter, the Fourier transform derivative property, which will be discussed in more detail in the Conclusion and Recommendation section, should be implemented.

0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	1	1	0	0
0	0	0	0	1	1	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Pattern Shifted
(20 urad, 20 urad)

0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	1	1	0	0	0
0	0	0	1	1	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Pattern Resulting From Shift Negation

Figure 7. Results of Shift Routine

0	0	0	0	0	0	0	0
0	0	0	0.455	0.882	0	0	0
0	0	0	0.882	1.71	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0.455	0.882	0	0	0.455	0.882	0
0	0.882	1.71	0	0	0.882	1.71	0
0	0	0	0	0	0	0	0

Figure 8. No Noise $\sigma = 1/4$ pixel $\alpha = 0.1$

0.352	0.283	-0.649	0.519	-0.287	-0.161	-0.427	-0.229
-0.382	-0.222	-0.798	-0.434	1.20	0.263	1.023	0.9622
-0.05	0.287	-0.355	1.53	1.61	1.29	0.138	1.22
-0.156	-0.383	1.71	1.12	0.221	1.26	0.416	1.155
0.731	0.843	-0.042	0.645	-0.176	0.303	1.31	-0.095
-0.082	0.135	-0.323	-0.401	-0.196	-0.336	0.5	-0.129
0.361	0.091	0.36	0.568	1.733	-0.631	1.28	-0.557

Figure 9A. 1st Run σ = 1/4 Pixel S/N = 10 α = 0.1

0.10	0.136	0.0318	0.022	-0.149	-0.218	0.0556	-0.0521
-0.159	-0.232	-0.294	0.444	0.349	0.133	0.183	-0.129
-0.153	0.158	0.684	1.049	0.799	0.223	0.724	0.641
0.087	0.279	0.738	0.572	-0.071	0.158	0.353	0.148
-0.113	-0.131	-0.045	0.127	0.011	0.135	-0.086	0.413
-0.129	0.117	0.319	-0.364	-0.242	-0.121	0.270	-0.081
-0.052	0.398	0.752	0.137	1.148	0.917	1.334	-0.435
0.218	0.293	-0.109	-0.235	0.643	0.133	-0.155	0.133

Figure 9B. 50th Run σ = 1/4 Pixel S/N = 10 α = 0.1

0.352	0.283	-0.649	-0.520	-0.287	-0.161	-0.427	-0.229
-0.382	-0.222	-0.798	-0.207	1.642	0.263	1.023	0.962
-0.049	0.287	0.499	2.41	2.46	1.29	0.992	1.66
-0.156	-0.383	2.15	1.35	0.221	1.26	0.857	1.38
0.731	0.843	-0.042	0.645	-0.176	0.303	1.31	-0.095
-0.083	0.363	1.175	-0.401	-0.196	-0.108	0.941	-0.129
0.361	0.532	1.21	0.568	2.59	0.250	2.14	-0.557
0.297	-0.485	0.378	0.098	2.22	-0.425	-0.467	0.674

Figure 10A. 1st Run $\sigma = 1/4$ Pixel S/N = 20 $\alpha = 0.1$

0.10	0.136	0.032	0.022	-0.149	-0.218	0.056	-0.052
-0.159	-0.231	-0.293	0.683	0.814	0.134	0.183	-0.129
-0.153	0.158	1.49	1.93	1.70	0.223	1.53	1.06
0.087	0.279	1.16	0.787	-0.071	0.158	0.77	0.363
-0.113	-0.131	-0.045	0.127	0.011	0.135	-0.086	0.413
-0.129	0.357	0.784	-0.364	-0.242	0.118	0.734	-0.081
-0.052	0.862	1.65	0.138	1.96	1.80	2.23	-0.434
0.218	0.293	-0.109	-0.235	1.06	0.348	-0.154	0.133

Figure 10B. 50th Run $\sigma = 1/4$ Pixel S/N = 20 $\alpha = 0.1$

12.17	16.46	20.22	22.51	22.73	20.81	17.29	13.03
16.04	21.43	26.08	28.90	29.17	26.82	22.46	17.12
19.84	26.14	31.49	34.72	35.02	32.34	27.33	21.11
23.06	29.97	35.73	39.17	39.49	36.64	31.25	24.47
25.13	32.24	38.08	41.51	41.84	38.99	33.55	26.60
25.52	32.39	37.94	41.16	41.47	38.79	33.64	26.94
23.95	30.14	35.07	37.92	38.18	35.83	31.26	25.23
20.61	25.78	29.86	32.18	32.40	30.48	26.71	21.69

Figure 11. No Noise IMAX = 20 σ = 3 pixels

5.51	7.51	7.71	11.22	11.11	10.36	8.79	6.81
8.16	11.49	12.90	13.33	15.83	13.81	10.42	9.17
9.60	13.16	15.75	15.41	18.81	16.96	11.75	9.83
10.51	15.21	18.73	19.04	19.77	19.11	14.41	13.36
14.14	15.93	20.27	20.42	20.80	20.38	16.31	13.03
13.80	16.33	19.86	21.89	20.67	21.17	17.56	12.62
12.0	15.0	17.91	20.10	19.82	19.27	15.13	13.06
11.31	13.51	16.26	16.04	16.10	13.79	10.31	10.69

Figure 12A. 1st Run σ = 3 pixels S/N = 10 α = 0.1

3.304	4.472	5.355	5.951	5.838	5.264	4.608	3.378
4.064	10.55	12.89	15.03	15.24	14.81	13.16	10.48
5.071	13.01	15.57	17.79	18.16	17.71	15.41	12.92
6.16	14.55	17.69	19.85	20.14	19.55	17.14	14.04
6.51	14.66	17.92	20.29	20.93	20.23	17.76	15.05
6.59	14.20	17.24	19.15	20.03	19.12	17.12	14.11
6.25	12.87	15.56	17.78	18.68	17.69	16.12	12.40
5.65	11.13	13.07	14.59	15.66	14.78	13.05	10.92

Figure 12B. 50th Run σ = 3 pixels S/N = 10 α = 0.1

11.59	15.74	17.82	22.48	22.47	20.77	17.44	13.32
16.17	22.20	25.94	27.79	30.41	27.22	21.66	17.73
19.52	26.23	31.50	32.77	36.32	33.13	25.41	20.39
22.04	30.19	36.60	38.62	39.51	37.43	30.04	25.60
26.70	32.05	39.31	41.18	41.72	39.88	33.09	26.32
26.55	32.53	38.83	42.47	41.40	40.56	34.37	26.09
23.97	30.06	35.44	39.06	38.91	37.18	30.76	25.68
21.61	26.40	31.19	32.13	32.30	29.03	23.66	21.53

Figure 13A. 1st Run σ = 3 pixels S/N = 20 α = 0.1

6.51	8.81	10.67	11.87	11.82	10.74	9.16	6.81
8.29	21.32	26.08	29.85	30.59	29.49	26.14	21.09
10.29	25.87	31.26	35.41	36.42	35.20	30.91	25.61
12.23	28.81	35.06	39.35	40.36	38.94	34.35	28.15
13.12	29.44	35.89	40.45	41.85	40.32	35.61	29.83
13.31	28.52	34.63	38.66	40.29	38.61	34.43	28.29
12.56	25.80	31.26	35.41	37.03	35.34	31.80	25.23
11.07	21.97	26.24	29.41	31.10	29.65	26.26	21.70

Figure 13B. 50th Run σ = 3 pixels S/N = 20 α = 0.1

Measurement Update No.	Correct Results		Unknown h Functions		Known Non-linear h Function		Known h Functions	
	x_D	y_D	x_D	y_D	x_D	y_D	x_D	y_D
1	0.0	0.0	-0.565	0.934	-0.0028	0.449	0.00137	0.4178
5	0.0	0.0	***	8.307	-4.82	0.126	-4.608	0.0163
10	0.0	0.0	***	57.09	***	***	***	***
20	0.0	0.0	***	***	***	***	***	***
50	0.0	0.0	***	***	***	***	***	***

*** Absolute value exceeds 100

Table 1. Divergence Analysis

IV Conclusions and Recommendations

In conclusion, this thesis did not accomplish nearly as much as anticipated at the beginning of the project. However, a number of milestones were realized. To name a few, the digital implementation of a negating phase shift, the validation of the FOURT subroutine, and the implementation of the exponential smoothing technique were highlights of this project. Since the investigation of the deterministic stationary truth model identified a possible problem with the forward-backwards difference approximation, the deterministic constant velocity and stochastic truth models were not analyzed.

Although some analysis was accomplished with this project, more is needed. To begin, the arbitrary assumption of padding the input array with 8 additional rows and columns of zeroes could be investigated in more detail to determine if little distortion results in using less zeroes. In addition, a further extension on the analysis of the averaging technique would involve not only a further investigation of its dependency on the image shape, but also an investigation of how the filter's performance is affected when the steady state value of the smoothing constant α is varied. Another modification to the computer algorithm that could be made deals with the sampling scheme described in Appendix A. A more unbiased and resolvable sampling scheme could be implemented. However, the addition of the Fourier transform derivative property would be the most important modification to the existing computer software, since it provides the best hope of solving the filter divergence problem. Analytically, the spatial derivative of the nonlinear h function represents the linearized h function. This spatial

derivative could be more precisely calculated by using the following Fourier property:

$$T\left\{\frac{\partial h(x,y)}{\partial x}\right\} = +j2\pi fx\ H(fx,fy) \qquad (63)$$

$$T\left\{\frac{\partial h(x,y)}{\partial y}\right\} = j2\pi fy\ H(fx,fy) \quad (17:314) \qquad (64)$$

As a possible plan of attack for future research, the following steps may be taken. First and most importantly, the filter divergence problem must be solved, hopefully with the use of the Fourier Transform divergence property. Next, the deterministic constant velocity and stochastic truth models should be employed keeping in mind that the normal robustness that is characteristic of Kalman filters may not be possible since the negating phase shift requires extremely accurate estimates of the target location. Once this has been performed, the filter performance can be evaluated based on changes to the zero padding of the input array, changes to the sampling scheme of the same input array, and variation to the steady state smoothing constant.

Bibliography

1. Bass M., Copley S., Beck D. G., and Wallace R. J. "Laser Assisted Hot Spot Maching," Proceeding of the Symposium on Laser-Solid Interactions and Laser Processing, 205-11, Boston, Massachusetts, November 27 - December 1, 1978.

2. Pertsov, O. L. "Biophysical Aspects of the Use of Lasers in Medical Research," Soviet Journal of Optical Technology, 46: 360-66 (June 1979).

3. Mercier, Daniel E. "An Extended Kalman Filter for Use in a Shared Aperture Medium Range Tracker," M.S. Thesis, Air Force Institute of Technology, Wright-Patterson AFB, Ohio, December 1978.

4. Maybeck, Peter S. "The Kalman Filter an Introduction for Potential Users," June 1972.

5. Harnly, Douglas A. and Jensen, Robert L. "An Adaptive Distributed-Measurement Extended Kalman Filter for a Short Range Tracker," M.S. Thesis, Air Force Institute of Technology, Wright-Patterson AFB, Ohio, December 1979.

6. Hanuise, Guy. "Image Processing by Computer," 1975.

7. Goodman, Joseph W. Introduction to Fourier Optics. McGraw-Hill Book Company, 1968.

8. Fukunaga, Keinosuke. Introduction to Statistical Pattern Recognition. New York: Academic Press Incorporated, 1972.

9. Oppenheim, Alan V. and Schafer, Ronald W. Digital Signal Processing. New Jersey: Prentice-Hall Incorporated, 1975.

10. Cooley J. W., Lewis P. A., and Welch P. D. "Historical Notes on the Fast Fourier Transform," IEEE Transactions on Audio and Electroacoustics, Volume Au-15 Number 2: 76-79 (June 1967).

11. Bush, Larry F. "The Design of an Optimum Alphanumeric Symbol Set for Cockpit Displays," M.S. Thesis, Air Force Institute of Technology, Wright-Patterson AFB, December 1977.

12. Bedworth, David D. Industrial Systems Planning, Analysis, Control. New York: The Ronald Press Company, 1973.

13. Maybeck, Peter S., Professor of Electrical Engineering. Personal interview. Air Force Institute of Technology, Wright-Patterson AFB, Ohio, August 1, 1980.

14. The Analytic Sciences Corporation. Advanced Adaptive Optics Control Techniques. TR-996-1. Prepared for the Air Force Weapons Laboratory, Kirtland Air Force Base, New Mexico, January 6, 1978.

15. Hogge, C. B. and Butts, R. R. "Frequency Spectra for the Geometric Representation of Wavefront Distortions Due to Atmospheric Turbulence," IEEE Transactions on Antenna and Propagation, Vol. AP-24, No. 2, March 1976. (Program supplied by authors).

16. Maybeck, Peter S. Stochastic Models, Estimation, and Control Volume I. New York: Academic Press Incorporated, 1979.

17. Gaskill, Jack D. Linear Systems, Fourier Transforms, and Optics. John Wiley and Sons, 1978.

18. Hornbeck, Robert W. Numerical Methods. Quantum Publishers, Incorporated, 1975.

APPENDIX A

Numerical Approximation

In the development of this thesis project, two numerical approximations were used. In generating the measurement vector $\underline{z}(t_i)$, the first approximation involved computing the average intensity over each pixel. Algebraically, the component of the $\underline{z}(t_i)$ vector corresponding to a particular pixel is

$$z_{jk}(t_i) = \frac{1}{A_p} \iint_{\substack{\text{region of} \\ jk^{th} \text{ pixel}}} I_{target}(x,y,t_i)\,dx\,dy + V_{jk}(t_i) \tag{65}$$

where A_p = area of one pixel

$I(x,y,t_i)$ = two-dimensional intensity pattern

$v_{jk}(t_i)$ = noise effect on the jk^{th} pixel

$z_{jk}(t_i)$ = actual measurement from jk^{th} pixel

Since a simple functional form of the intensity pattern that can be analytically integrated is not available, the exact integration of equation 65 cannot be performed. Instead, 25 sample points from each pixel will be taken and averaged to provide the average intensity over the pixel. The exact locations are given in Figure 14 below. It should be noted that samples are taken along the top and left edge of the pixel of interest so that duplicate sampling does not occur along the edges. In other words, the right edge becomes the sampled left edge for the next pixel to the right, while the bottom edge becomes the sampled top edge for the pixel just below. The selection of this sampling scheme was arbitrary. In follow-on projects, more samples from different locations may be taken to provide better resolution and thus more sensitivity

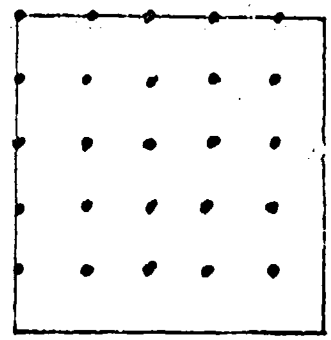

Figure 14. Pixel Sampling Scheme

to the movement of the intensity pattern from one sample period to another.

The second approximation, which is indirectly tied to the first, involves the generation of the linearized h function. Since an explicit analytical form of the non-linear h function is not available, an exact differentiation of this non-linear function to provide a linearized function cannot be performed. Instead, a numerical approximation known as the Forward-Backwards Difference Method is used (18:16-17). A familiar numerical method used in finite difference calculus, this technique is used to compute the derivative of a function to any higher order. However, to compute the first order spatial derivative for a particular pixel, the pixel values, along the direction for which the spatial derivative is taken, just before and just after the pixel of interest is used. The values are subtracted from each other and divided by the distance between each other (i.e. 40 urad). For the pixels located along the edges for which the pixel just before or just after does not exist, either just a forward or just a backwards difference is taken. To be more specific, the value of the pixel itself is subtracted from

either the value of the pixel just before or just after, whichever is available, and divided by the distance between the two (i.e. 20 urad). For example, referring to Figure 5, Pixel Numbering Scheme, if the spatial derivative in the x direction for pixel number 29 is to be computed, the value for pixel 28 is subtracted from the value for pixel 30 and divided by 40. However, along the edge, just a forward or backward difference has to be calculated. For instance, to find the spatial derivative in the y direction for pixel number 3, the value for pixel 3 is subtracted from the value for pixel 11 and divided by 20.

APPENDIX B

Coordinate System for 8 x 8 Input Array

The development of a coordinate system to use throughout this project was critical. The 8 x 8 input array represents an 8 x 8 FLIR array whose individual pixel FOV is 20 urad x 20 urad and system FOV is 160 urad x 160 urad. With the additional 8 rows and columns of zeroes (see Fourier Transform section), the resulting 24 x 24 array of data actually represents a coordinate system that is 480 urad x 480 urad with the (0,0) coordinate in the upper left hand corner, x increasing from left to right and y increasing from top to bottom (Figure 15).

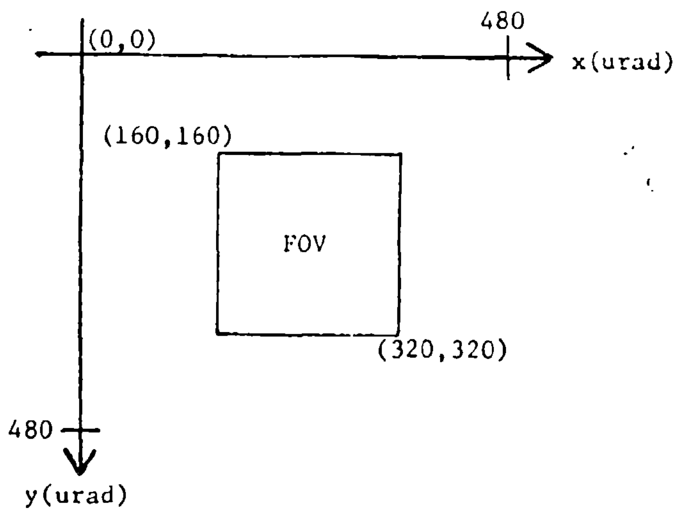

Figure 15. Coordinate System for Input Array

APPENDIX C

Input Patterns

Although the development of the Extended Kalman Filter for this thesis assumed no apriori knowledge concerning the exact algebraic form of the intensity pattern, three different intensity patterns were developed to provide the measurement vector $\underline{z}(t_i)$ necessary to verify the performance of the Kalman Filter. The actual computer subroutines (Input 1, Input 2, Input 3) are described in Appendix D. However, the three input patterns are a 40 x 40 constant intensity block, 3 constant height cylinder patterns, and 3 gaussian profiles. Figures 16a,b,c illustrate how the centered patterns would appear on the FLIR array. Figures 17a,b,c show the resulting noise-free values for the average intensity per pixel. The selection of these three patterns is significant, since the 40 x 40 pattern provides a very pronounced difference between zero and non-zero values, while the 3 cylinder pattern once again provides a pronounced difference between zero and non-zero values but is closer to simulating a multiple-hot-spot target, while the 3 gaussian profiles provide the best representation of the multiple-hot-spot targets.

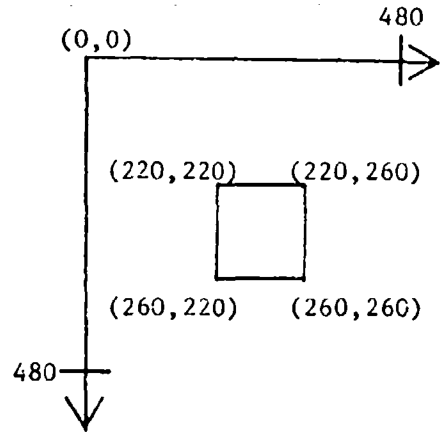

Figure 16a. 40 x 40 Constant Intensity

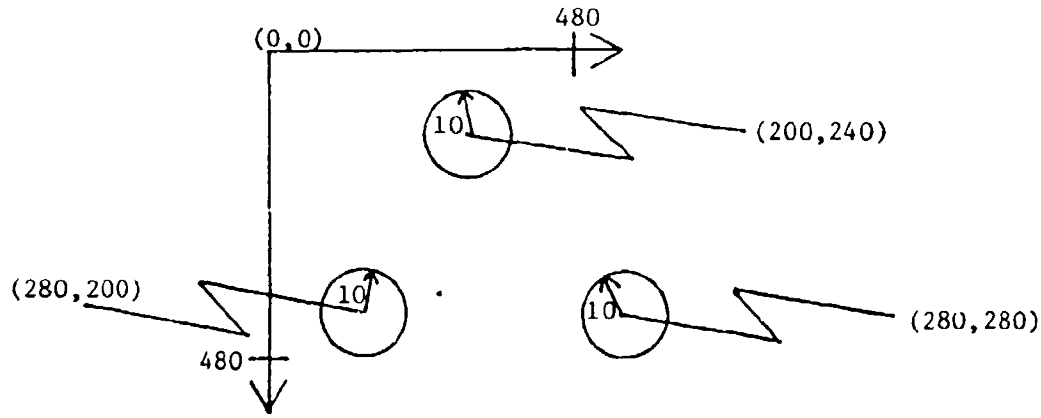

Figure 16b. 3 Constant Height Cylinders

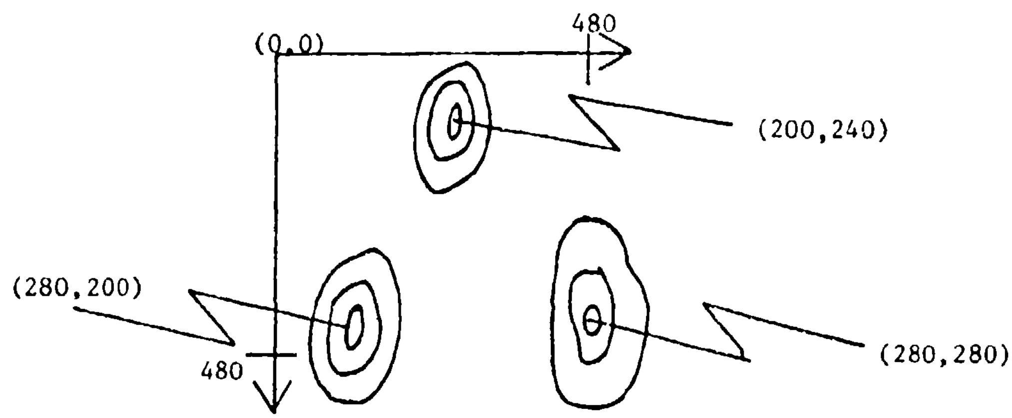

Figure 16c. 3 Gaussian Profiles

0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	1	1	0	0	0
0	0	0	1	1	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Figure 17a. Discrete 40 x 40 Constant Intensity

0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0.2	0.32	0	0	0
0	0	0	0.12	0.2	0	0	0
0	0	0	0	0	0	0	0
0	0.2	0.32	0	0	0.2	0.32	0
0	0.12	0.2	0	0	0.12	0.2	0
0	0	0	0	0	0	0	0

Figure 17b. Discrete 3 Constant Height Cylinders

0	0	0	0	0	0	0	0
0	0	0	0.045	0.0881	0	0	0
0	0	0	0.0881	0.171	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0.045	0.0881	0	0	0.045	0.0881	0
0	0.0881	0.171	0	0	0.0881	0.171	0
0	0	0	0	0	0	0	0

$\sigma = 1/4$ Pixel

Figure 17c. Discrete 3 Gaussian Profiles

APPENDIX D

Computer Software

This appendix contains the Fortran computer software used for this project. The computer software only reflects the use of the stationary target truth model. Since some models were used from previous thesis, there may be some duplication of their work within this software. This program was written for use on the CDC 6600 Fortran IV compiler.

```
C** *THIS PROGRAM COMPUTES FFT AND EXPONENTIAL SMOOTHING (A=1.1)
      PROGRAM MAIN(INPUT,OUTPUT,TAPE5=INPUT,TAPE6=OUTPUT,TAPE5,
     1 TAPE6=OUTPUT)
      DIMENSION DATA1(24,24),DATA2(24,24),DATA3(4,4),NN(2),
     1 WORK( ),S(12),IMAX(3),C( ),P(24,24),PP(4,4),PHI(4,4),
     2 PI(4,4),V( ),PHIN(64,64),XX(4),YY(4),NLH( ),LH( , ),
     3 TAV( , ),Z( ),PI( , ),W(64,54)
      COMPLEX DATA1,DATA2,XX,YY
      COMPLEX WORK
      REAL IMAX,KK,NLH,LH
C** *INITIALIZATION OF VALUES
      CALL RANSET(123)
      K =
      KOUNT1=
      KOUNT2=
      DF1= .
      DF2= .
      N=24
      KK= .
      NN(1)=24
      NN(2)=24
      NDIM=2
      ITOT=NN(1)*NN(2)
      IFORM=1
      DO    I=1,4
      DO    J=1,4
      PP(I,J)= .
      PHI(I,J)=
      PI(I,J)=
   33 CONTINUE
      DO    I=1,N
      DO    J=1,N
      DATA2(I,J)= .
   37 CONTINUE
      DO 1  I=1,64
      V(I)= .
      DO 1  J=1,
      PIJ(I,J)=
      (I,J)=
   13 CONTINUE
C** *IDENTIFICATION OF INPUT PATTERN
      READ*,IP
      IF(IP .EQ.1)GO TO 31
      IF(IP .EQ.2)GO TO 32
      IF(IP .EQ.3)GO TO 33
      PRINT*,"ERROR IN IDENTIFICATION OF INPUT PATTERN"
      GO TO 99
   31 READ*,IMAX(1)
   91 FORMAT(1X,"THIS PROGRAM A X X",F5.2,
     1 1X,"INTENSITY PATTERN")
      PRINT 91,IMAX(1)
      CALL INPUT1(DATA1,DF1,DF2,N,IMAX,DENX,DENY)
      DENOX=DENX
      DENOY=DENY
      GO TO 4
      READ*,IMAX(1),IMAX(2),IMAX(3)
```


Best Available Copy.

```
      1 " THE 3 CYLINDER INTENSITY PATTERN",/,
      2 1X,"THE HEIGHT OF CYLINDER 1 IS",1X,F5.2,/,
      3 1X,"THE HEIGHT OF CYLINDER 2 IS",1X,F5.2,/,
      4 1X,"THE HEIGHT OF CYLINDER 3 IS",1X,F5.2)
      PRINT 92,IMAX(1),IMAX(2),IMAX(3)
      CALL INPUT2(DATA1,OF1,OF2,N,IMAX,DENX,DENY)
      CTDDY=DENX
      DTDDY=DENY
      GO TO 36
 33   READ*,IMAX(1),IMAX(2),IMAX(3)
 93   FORMAT(1X,"THIS PROGRAM WILL USE",
      1 " THE 3 GAUSSIAN PROFILE INTENSITY PATTERN",/,
      2 1X,"GAUSSIAN PROFILE",1X,I1,1X,"HAS A HEIGHT OF",1X,
      3 F5.2,/,1X,"WITH THE ELEMENTS OF ITS INVERSE",
      4 " COVARIANCE MATRIX BEING",4(1X,F10.7),/)
      DO 13 I=1,3
      I1= (I-1)*4 + 1
      I2= (I-1)*4 + 2
      I3= (I-1)*4 + 3
      I4= (I-1)*4 + 4
      READ*,S(I1),S(I2),S(I3),S(I4)
      PRINT 93,I,IMAX(I),S(I1),S(I2),S(I3),S(I4)
 13   CONTINUE
      CALL INPUT3(DATA1,OF1,OF2,N,3,IMAX,DENX,DENY)
      CTDDY=DENX
      DTDDY=DENY
 34   READ*,NRUNS
 35   FORMAT(1X,/////,1X,"THIS PROGRAM WILL DO",
      1 1X,I3,1X,"RUNS IN ITS MONTE CARLO SIMULATION.",//)
      PRINT 35,NRUNS
      READ*,DELT
      PRINT*,"TIME BETWEEN SAMPLE MEASUREMENTS=",DELT
      READ*,SIGMAD
      PRINT*,"RMS TARGET DYNAMICS=",SIGMAD
      READ*,TAUD
      PRINT*,"CORRELATION TIME FOR TARGET DYNAMICS=",TAUD
      READ*,SIGMAA
      PRINT*,"RMS FOR ATMOSPHERIC MODEL=",SIGMAA
      READ*,TAUA
      PRINT*,"CORRELATION TIME FOR ATMOSPHERIC MODEL=",TAUA
      READ*,RINV
      PRINT*,"KALMAN FILTER VARIANCE VALUE=",RINV
      READ*,XP(1),XP(2),XP(3),XP(4)
      PRINT*,"INITIAL STATE VECTOR"
      PRINT*," XD=",XP(1)
      PRINT*," YD=",XP(2)
      PRINT*," XA=",XP(3)
      PRINT*," YA=",XP(4)
      READ*,PP(1,1),PP(2,2),PP(3,3),PP(4,4)
      PRINT*,"INITIAL COVARIANCE MATRIX"
      DO 1 I=1,4
      PRINT*,(PP(I,J),J=1,4)
 1    CONTINUE
C**** IDENTIFICATION OF TYPE NOISE
      READ*,SN
      IF(SN.EQ.0.) GO TO 72
      IF(SN.EQ.1.) GO TO 74
```

Best Available Copy

```
      IF(SN.EQ.2.0)GO TO 72
      IF(SN.EQ.3.0)GO TO 73
      PRINT*,"ERROR IN INDENTIFING NOISE TYPE",
     1" WILL ASSUME NO NOISE CORRUPTION IN SIMULATION"
      SN=1.0
      GO TO 79
C***GENERATION OF SPATIAL NOISE CORRELATION COEFFICIENT MATRIX
71    READ*,SIGMAS
      PRINT*,"RMS SPATIAL NOISE=",SIGMAS
      READ*,C(1),C(2),C(3),C(4),C(5)
      PRINT*,"SPATIAL NOISE CORRELATION COEFFICIENTS"
      PRINT*,C(1),C(2),C(3),C(4),C(5)
      CALL SPIN(C,SIGMAS,R,RN)
      CALL NOISE(FATA1,R,SN,V,PHIN)
      GO TO 79
C***GENERATION OF TEMPORAL NOISE CORRELATION MATRIX
72    READ*,SIGMAT
      PRINT*,"RMS TEMPORAL NOISE=",SIGMAT
      READ*,TAUN
      PRINT*,"CORRELATION TIME FOR TEMPORAL NOISE=",TAUN
      ARG=-DELT/TAUN
      SIGMAT=SIGMAT*(1.0-EXP(2.0*ARG))
      DO 75 I=1,4
      PHIN(I,I)=EXP(ARG)
75    CONTINUE
      C(1)=1.0
      C(2)=0.
      C(3)=0.
      C(4)=0.
      C(5)=0.
      CALL SPIN(C,SIGMAT,R,RN)
      CALL NOISE(FATA1,R,SN,V,PHIN)
      GO TO 79
C***GENERATION OF SPATIAL AND TEMPORAL NOISE CORRELATION MATRIX
73    READ*,SIGMAR
      PRINT*,"RMS SPATIAL AND TEMPORAL NOISE",SIGMAR
      READ*,C(1),C(2),C(3),C(4),C(5)
      PRINT*,"NOISE CORRELATION COEFFICIENTS"
      PRINT*,C(1),C(2),C(3),C(4),C(5)
      READ*,TAUN
      PRINT*,"NOISE CORRELATION TIME=",TAUN
      ARG=-DELT/TAUN
      SIGMAR=SIGMAR*(1.0-EXP(2.0*ARG))
      DO 77 I=1,4
      PHIN(I,I)=EXP(ARG)
77    CONTINUE
      CALL SPIN(C,SIGMAT,R,RN)
      CALL NOISE(FATA1,R,SN,V,PHIN)
C***COMPUTATION OF STATE TRANSITION MATRIX
79    PHI(1,1)=EXP(-DELT/TAUO)
      PHI(2,2)=PHI(1,1)
      PHI(3,3)=EXP(-DELT/TAUA)
      PHI(4,4)=PHI(3,3)
      PRINT*,"STATE TRANSITION MATRIX"
      DO 21 I=1,4
      PRINT*,(PHI(I,J),J=1,4)
21    CONTINUE
```

```
      DD(1,1)=12.
      DD(2,2)=DD(1,1)
      DD(3,3)=SIGF**(1.-EXP(-2.*DELT/TAU))
      DD(4,4)=DD(3,3)
      PRINT *,"DD MATRIX"
      DO 22 I=1,4
      PRINT *,(DD(I,J),J=1,4)
 22   CONTINUE
 21   K=K +1
      IF(K.GT.1)GO TO 25
      SHFX=?-KX-CENX
      SHFY=?NY-CENY
      GO TO 26
 25   SHFX=XP(1) + XP(3)
      SHFY=XP(2) + XP(4)
 26   CALL ACC(SHFX)
      CALL ACC(SHFY)
      ISIGN=1
      CALL FOURT(DATA1,NN,NDIM,ISIGN,IFORM,WORK)
C** SPATIAL SHIFT TO CENTER H FUNCTION
      CALL SHIFT(DATA1,SHFX,SHFY)
 31   FORMAT(1H ,6(2X,1PE10.3))
C** COUNTER FOR EXPONENTIAL SMOOTHING
      KK=KK + 1
      IF (KK.GT.?) GO TO ?
      A=1/KK
C** AVERAGING WITH PAST DATA USING EXPONENTIAL SMOOTHING
 53   DO 1? I=1,N
      DO 1 J=1,N
      DATA2(I,J)= (A*DATA1(I,J)) + ((1.0-A)*DATA2(I,J))
 1    CONTINUE
C** INVERSE FFT
      ISIGN=-1
      CALL FOURT(DATA2,NN,NDIM,ISIGN,IFORM,WORK)
      DO 52 I=1,N
      DO 52 J=1,N
      DATA2(I,J)=DATA2(I,J)/TOT
C** ANYVALUE LESS THAN 1.**-5 IS CONSIDERED ZERO
      R=REAL(DATA2(I,J))
      R=ABS(R)
      IF(R.LT.1.?E-?)DATA2(I,J)=?.
 52   CONTINUE
C** SECOND CHECK
      PRINT*,"AVERAGED IMAGE"
      DO ? I=1,N
      PRINT ?,(DATA2(I,J),J=1,N)
 ?1   CONTINUE
C** GENERATION OF NONLINEAR H FUNCTION FROM MIDDLE
C     8X8 OF DATA2
      K=?
      DO 15 I=?,1?
      DO 15 J=?,1?
      K=K + 1
      HLH(K)=DATA2(I,J)
      DATA2(I-?,J-?)=DATA2(I,J)
 15   CONTINUE
      PRINT*,"NONLINEAR H FUNCTION"
```

```
      PRINT 3.,(ULH(K),K=1,64)
C** *GENERATION OF LINEARIZED H-FUNCTION
      CALL LHF(DATA3,LH)
C** *SAME AVERAGING TECHNIQUE IS DONE WITH FOURIER TRANSFORM,
C     THE FFT OF DATA3 WILL BE STORED IN MEMORY
      ISIGN=1
      CALL FOURT(DATA3,NN,NDIM,ISIGN,IFORM,WORK)
C** *PROPAGATION OF STATE AND COVARIANCE MATRIX
      CALL PROP(XF,PF,PHI,QD,XM,PM)
C** *GENERATION OF INPUT ARRAY
      OF1=-XM(1)
      OF2=-XM(2)
      PR1=OF1
      PR2=OF2
      CALL ACC(OF1)
      CALL ACC(OF2)
      PRINT*,"OF1=",OF1
      PRINT*,"OF2=",OF2
      IF(IP.EQ.1)GO TO 51
      IF(IP.EQ.2)GO TO 52
      IF(IP.EQ.3)GO TO 63
51    CALL INPUT1(DATA1,OF1,OF2,N,IMAX,CENX,CENY)
      CALL NOISE(DATA1,R,SN,V,PHIN)
      GO TO 54
52    CALL INPUT2(DATA1,OF1,OF2,N,IMAX,CENX,CENY)
      CALL NOISE(DATA1,R,SN,V,PHIN)
      GO TO 54
53    CALL INPUT3(DATA1,OF1,OF2,N,S,IMAX,CENX,CENY)
      CALL NOISE(DATA1,R,SN,V,PHIN)
C** *GENERATION OF MEASUREMENT VECTOR Z
      OF1=PR1
      OF2=PR2
54    K=
      DO 37 I=5,11
      DO 37 J=5,11
      K=K+1
      Z(K)=DATA1(I,J)
37    CONTINUE
C** *SECOND CHECK OF MEASUREMENT VECTOR Z
      PRINT*,"Z VECTOR"
      PRINT 30,(Z(K),K=1,64)
C** *PERFORMANCE ANALYSIS
      KOUT1=KOUT1 + 1
      CALL ANLY(XM,CENX,CENY,KOUT1,SAV)
C** *MEASUREMENT UPDATE OF STATE VECTOR AND COVARIANCE MATRIX
      CALL UPD(Z,FLH,LH,XM,PM,RINV,XP,PP)
      KOUT2=KOUT2 + 1
      PRINT*,"KOUT2=",KOUT2
      IF(KOUT2.LT.NRUNS)GO TO 23
C** *RESULTS OF PERFORMANCE ANALYSIS DISPLAYED
31    FORMAT(20X,"RESULTS OF MONTE CARLO SIMULATION",////,
     1 1X,"RUN #",1X,"X MINUS",13X,"CENX",13X,"DIFF",12X,
     2 "Y  TRUE",11X,"CENY",13X,"DIFF",//)
31    FORMAT(6X,I3,1(2X,1PE15.8),/)
      PRINT 3.
      DO 32 I=1,NRUNS
      PRINT 31,I,SAV(I,1),SAV(I,2),SAV(I,5),SAV(I,7),
```

```
      1 SAV(I,4),SAV(I,5)
   32 CONTINUE
C*** *COMPUTATION OF MEAN AND STANDARD DEVIATION
      X2=0.0
      Y2=0.0
      XMEAN=0.
      YMEAN=0.
      DO 34 I=1,NRUNS
      XMEAN=XMEAN + SAV(I,5)
      YMEAN=YMEAN + SAV(I,6)
      X2=X2 + SAV(I,5)**2
      Y2=Y2 + SAV(I,6)**2
   34 CONTINUE
      C3=NRUNS
      XMEAN=XMEAN/C3
      YMEAN=YMEAN/C3
      SX=((C3*X2) -(XMEAN**2))/(C3*(C3-1.0))
      SX=SQRT(SX)
      SY=((C3*Y2) - (YMEAN**2))/(C3*(C3-1.0))
      SY=SQRT(SY)
   35 FORMAT(1X,//,1X,"MEAN DIFFX=",1PE13.3,/,
     1 1X,"STD DEVIATION OF DIFFX=",1PE13.3,/,
     2 1X,"MEAN DIFFY=",1PE13.3,/,
     3 1X,"STD DEVIATION OF DIFFY=",1PE13.3)
      PRINT 35,XMEAN,SX,YMEAN,SY
   33 STOP"END OF PROGRAM"
      END
```

SYMBOLIC REFERENCE MAP (R=1)

ENTRY POINTS
6217 MAIN

VARIABLES	SN	TYPE		RELOCATION				
14255	A	REAL			14253	ARG	REAL	
14251	B	REAL			14253	BB1	REAL	
14254	BB2	REAL			15151	C	REAL	ARRAY
14232	DENOM	REAL			14237	DENOY	REAL	
14230	DENX	REAL			14231	DENY	REAL	
14241	D3	REAL			12074	DATA1	COMPLEX	ARRAY
12474	DATA2	COMPLEX	ARRAY		14474	DATA3	REAL	ARRAY
14261	DELT	REAL			14223	I	INTEGER	
14224	IFORM	INTEGER			15155	IMAX	REAL	ARRAY
14227	IP	INTEGER			14257	ISIGN	INTEGER	
14234	I1	INTEGER			14235	I2	INTEGER	
14236	I3	INTEGER			14237	I4	INTEGER	
14220	J	INTEGER			14222	K	INTEGER	
14213	KK	REAL			14215	KOUT1	INTEGER	
14216	KOUT2	INTEGER			14214	KW	INTEGER	
35435	LM	REAL	ARRAY		14221	L	INTEGER	
14222	NDIM	INTEGER			34356	NLM	REAL	ARRAY
14774	NN	INTEGER	ARRAY		14240	NRUNS	INTEGER	
16217	OF1	REAL			14221	OF2	REAL	

```
      SUBROUTINE INPUT1(DATA1,OF1,OF2,N,IMAX,CENX,CENY)
      DIMENSION DATA1(N,N),IMAX(7)
      REAL IMAX,LBX,LBY,LB
      COMPLEX DATA1
C** MODIFICATION TO PUT ZERO OFFSET ONTO BORESIGHT
      OF1P=OF1 + 2.  .
      OF2P=OF2 + 2.  .
      PRINT*,"OF1P=",OF1P
      PRINT*,"OF2P=",OF2P
C**  CENTROID LOCATION
      CENX=OF1P
      CENY=OF2P
      PRINT*,"CENX=",CENX
      PRINT*,"CENY=",CENY
C** DETERMINATION OF BOUNDS FOR 4X4  INTENSITY PATTERN
      LBX=OF1P-  .
      UBX=OF1P+  .
      LBY=OF2P-  .
      UBY=OF2P+  .
      PRINT*,"LBX=",LBX
      PRINT*,"UBX=",UBX
      PRINT*,"LBY=",LBY
      PRINT*,"UBY=",UBY
C**  LIMITS FOR BOUNDS IN A 8X8 PIXEL ARRAY
      LB=1  .
      UB=32 .
      IF(LBX.LT.LB)GO TO 30
      IF(LBX.GT.UB)GO TO 31
      IF(UBX.LT.LB)GO TO 31
      IF(UBX.GT.UB)GO TO 32
      IF(LBY.LT.LB)GO TO 33
      IF(LBY.GT.UB)GO TO 31
      IF(UBY.LT.LB)GO TO 31
      IF(UBY.GT.UB)GO TO 34
      GO TO 35
30    LBX=1  .
      GO TO 35
33    LBY=1  .
      GO TO 35
31    PRINT*,"TARGET HAS BEEN DRIVEN FROM PIXEL ARRAY."
      PRINT*,"WILL GO TO NEXT FRAME OF DATA"
      OF1=999.
      GO TO 2
32    UBX=32 .
      GO TO 35
34    UBY=32 .
C** GENERATION OF INPUT ARRAY
C****EACH PIXEL IS DIVIDED INTO 25 PIECES FOR INTEGRATION
35    M=5
      N=25
      DO 1 I=1,N
      DO 2 J=1,N
      AVE= .
      DO 3 K1=1,M
      DO 4 K2=1,M
C** COMPUTATION OF X AND Y VALUES
      Y=(I-1)*2. + (K1-1)*4.
```

```
                 X=(J-1)*2.0 + (K2-1)*4.0
  C***  EVALUATION OF F(X,Y)
 5               IF(X.LT.LBX)GO TO 5
                 IF(Y.LT.LBY)GO TO 5
                 IF(X.GE.UBX)GO TO 5
                 IF(Y.GE.UBY)GO TO 5
                 FXY=FMAX(1)
 60              GO TO 4
            5    FXY=0.
            4    AVG=AVG+FXY
                 CONTINUE
            3    CONTINUE
 7.              DATA1(I,J)= AVG/25.0
            2    CONTINUE
            1    CONTINUE
            20   RETURN
                 END
```

SYMBOLIC REFERENCE MAP (R=1)

```
ENTRY POINTS
   3  INPUT1

VARIABLES    SN  TYPE        RELOCATION
  25   AVG       REAL                          0   DENX     REAL               F.P.
   0   DENY      REAL             F.P.         0   DATA1    COMPLEX   ARRAY    F.P.
  252  FXY       REAL                         253  I        INTEGER
   0   FMAX      REAL     ARRAY   F.P.        254  J        INTEGER
  255  K1        INTEGER                      257  K2       INTEGER
  24   LBX       REAL                         262  LBY      REAL
  253  LBY       REAL                         262  N        INTEGER
   0   N         INTEGER          F.P.         0   OF1      REAL               F.P.
  245  OF1P      REAL                          0   OF2      REAL               F.P.
  246  OF2P      REAL                         251  UBX      REAL
  247  UBX       REAL                         250  UBY      REAL
  251  X         REAL                         260  Y        REAL

FILE NAMES      MODE
    OUTPUT      FMT

STATEMENT LABELS
   0  1                      0  2                 6  3
   0  4                    137  5               143  5
  150  20                   62  30               56  31
   74  32                   64  33               76  34
  1..  35

LOOPS  LABEL   INDEX    FROM-TO   LENGTH    PROPERTIES
  114   1       I        51 72     543             NOT INNER
  115   2.      J        52 71     518             NOT INNER
  117   3       K1       54 69     373             NOT INNER
  122   4       K2       55 68     219          OPT
```

```
      SUBROUTINE INPUT2(DATA1,OF1,OF2,N,IMAX,CENX,CENY)
      DIMENSION DATA1(N,N),IMAX(3)
      COMPLEX DATA1
      REAL IMAX,NUMX,NUMY
C** EACH PIXEL IS DIVIDED INTO 25 PIECES FOR INTEGRATION
      M=5
      OX1=2 . + OF1
      OY1=2 . + OF2
      OX2=2 . + OF1
      OY2=28 . + OF2
      OX3=28. + OF1
      OY3=28 . + OF2
C** ROTATION OF CENTERS
      OX1P=OX1
      OY1P=OY1
      OX2P=OX2
      OY2P=OY2
      OX3P=OX3
      OY3P=OY3
      NUMX=0.
      NUMY=0.
      AVG1=0.
      DO 1 I=1,N
      DO 2 J=1,N
      AVG= .
      DO 3 K1=1,M
      DO 4 K2=1,M
C****COMPUTATION OF X AND Y VALUES
      X=(I-1)*2. + (K1-1)*4.
      Y=(J-1)*2. + (K2-1)*4.
C** ROTATION OF X AND Y VALUES
      XP=X
      YP=Y
C** LIMITS FOR FXY ARRAY
      IF(XP.LT.1.  .) GO TO 9
      IF(XP.GE.32  .)GO TO 9
      IF(YP.LT.1.  .) GO TO 9
      IF(YP.GE.32  .) GO TO 9
C****EVALUATION OF FXY
      VAL=(XP-OX1P)**2 + (YP-OY1P)**2
      IF(VAL.LE.  .)GO TO 11
      VAL=(XP-OX2P)**2 + (YP-OY2P)**2
      IF(VAL.LE.  .)GO TO 12
      VAL=(XP-OX3P)**2 + (YP-OY3P)**2
      IF(VAL.LE.  .)GO TO 13
    9 FXY= .
      GO TO 6
   11 FXY=IMAX(1)
      GO TO 6
   12 FXY=IMAX(2)
      GO TO 6
   13 FXY=IMAX(3)
      GO TO 6
    6 AVG=AVG+FXY
C****COMPUTATION OF NUMERATOR FOR CENTROID LOCATION
      NUMX=NUMX + X*FXY
      NUMY=NUMY + Y*FXY
```

Best Available Copy

```
      4     CONTINUE
      3     CONTINUE
60          AVG1=AVG1 + AVG
            DATA1(I,J)=AVG/2 .
      2     CONTINUE
      1     CONTINUE
            CTMX= DUMMX/AVG1
60          CTMY= DUMMY/AVG1
C***  CHECK FOR NO IMAGE
            K=
            DO 7 I=1,N
            DO 3 J=1,N
70          IF(DATA1(I,J).EQ. [illegible]) K=K+1
      3     CONTINUE
      7     CONTINUE
            IF(K.LT. [illegible]) GO TO 20
            PRINT*,"TARGET HAS BEEN DRIVEN FROM PIXEL ARRAY."
70          PRINT*,"WILL GO TO NEXT FRAME OF DATA."
            CT=999.
     20     RETURN
            END
```

SYMBOLIC REFERENCE MAP (R=1)

ENTRY POINTS
3 INPUT2

```
VARIABLES      SN  TYPE         RELOCATION
 277  AVG          REAL                             234  AVG1      REAL
   0  CENX         REAL                 F.P.          0  CENY      REAL                F.P.
   0  DATA1        COMPLEX    ARRAY     F.P.        247  EXY       REAL
 235  I            INTEGER                            0  EMAX      REAL      ARRAY     F.P.
 236  J            INTEGER                          2[illegible]  K         INTEGER
 24[illegible]  K1           INTEGER                          2[illegible]1  K2        INTEGER
 217  M            INTEGER                            0  N         INTEGER             F.P.
 21[illegible]  DUMX         REAL                             215  DUMY      REAL
   0  DF1          REAL                 F.P.          0  DF2       REAL                F.P.
 22[illegible]  DX1          REAL                             225  DX1P      REAL
 222  DX2          REAL                             23[illegible]  DX2P      REAL
 22[illegible]  DX3          REAL                             232  DX3P      REAL
 22[illegible]  DY1          REAL                             22[illegible]  DY1P      REAL
 223  DY2          REAL                             2[illegible]1  DY2P      REAL
 22[illegible]  DY3          REAL                             233  DY3P      REAL
 24[illegible]  VAL          REAL                             243  X         REAL
 244  XP           REAL                             242  Y         REAL
 24[illegible]  YP           REAL

FILE NAMES         MODE
       OUTPUT      FMT

STATEMENT LABELS
   0  1                   3  2                   [illegible]  3
   0  4                 111  6                   1  7
```

Best Available Copy

```
      SUBROUTINE INPUT3(DATA1,OF1,OF2,M,N,S,IMAX,SUMX,SUMY)
      DIMENSION DATA1(M,N),S(12),IMAX(3)
      COMPLEX DATA1
      REAL IMAX,SUMX,SUMY
C** EACH FIELD IS DIVIDED INTO 25 PIECES FOR INTEGRATION
C** INITIALIZATION
      M=[illegible]
C** CONFIGURATION OF PEAKS
      X1P=[illegible]. +OF1
      Y1P=[illegible]. + OF2
      X2P=[illegible]. + OF1
      Y2P=[illegible]. + OF2
      X3P=[illegible]. + OF1
      Y3P=[illegible]. + OF2
C** ROTATION OF PEAKS
      X1PP=X1P
      Y1PP=Y1P
      X2PP=X2P
      Y2PP=Y2P
      X3PP=X3P
      Y3PP=Y3P
      SUMX=0.
      SUMY=0.
      AVG=0.
      DO 1 I=1,N
      DO 2 J=1,N
      AVG=0.
      DO 3 K1=1,M
      DO 4 K2=1,M
C** COMPUTATION OF X AND Y VALUES
      X=(I-1)*2[illegible]. + (K1-1)*4.
      Y=(J-1)*2[illegible]. + (K2-1)*4.[illegible]
C** ROTATION OF X AND Y VALUES
      XP=X
      YP=Y
C** LIMITS FOR SUMX & SUMY
      IF(XP.LT.1[illegible].) GO TO 9
      IF(XP.GT.32[illegible].) GO TO 9
      IF(YP.LT.1[illegible].) GO TO 9
      IF(YP.GT.32[illegible].) GO TO 9
C** EVALUATION OF FXY
      X1=XP-X1PP
      Y1=YP-Y1PP
      X2=XP-X2PP
      Y2=YP-Y2PP
      X3=XP-X3PP
      Y3=YP-Y3PP
      ARG1=-.5*((X1**2*S(1)) + (X1*Y1)*(S(2) + S(3)) + (Y1**2*S(4)))
      ARG2=-.5*((X2**2*S(5)) + (X2*Y2)*(S(6) + S(7)) + (Y2**2*S(8)))
      ARG3=-.5*((X3**2*S(9)) + (X3*Y3)*(S(10) + S(11)) + (Y3**2*S(12)))
C** TEST FOR LOWER BOUND FOR EXP FUNCTION
      IF(ARG1.LT.-675.) ARG1=-675.
      IF(ARG2.LT.-675.) ARG2=-675.
      IF(ARG3.LT.-675.) ARG3=-675.
      FXY = IMAX(1)*EXP(ARG1) + IMAX(2)*EXP(ARG2) + IMAX(3)*EXP(ARG3)
      GO TO 5
    9 FXY= .
```

Best Available Copy

```
      5       AVG=AVG+FXY
              MUMX=MUMX + X*FXY
              MUMY=MUMY + Y*FXY
      4       CONTINUE
      3       CONTINUE
              AVG1=AVG1 + AVG
              DATA1(I,J)=AVG/2...
      2       CONTINUE
      1       CONTINUE
C** COMPUTATION OF CENTROID
              CTRX= MUMX/AVG1
              CTRY= MUMY/AVG1
C** CHECK FOR NO IMAGE
              K=
              DO 1 I=1,N
              DO 3 J=1,N
              IF(DATA1(I,J).LE.1.0E-6) K=K + 1
      3       CONTINUE
      1       CONTINUE
              IF(K.LT.....) GO TO 2.
              PRINT ,"TARGET HAS BEEN DRIVEN FROM PIXEL ARRAY."
              PRINT*,"WILL GO TO NEXT FRAME OF DATA."
              DF1=-999.
     2.       RETURN
              END
```


SYMBOLIC REFERENCE MAP (R=1)

ENTRY POINTS
 3 INPUT3

VARIABLES	SN	TYPE		RELOCATION					
312	ARG1	REAL			313	ARG2	REAL		
314	ARG3	REAL			275	AVG	REAL		
272	AVG1	REAL				DENX	REAL		F.P.
0	DENY	REAL		F.P.		DATA1	COMPLEX	ARRAY	F.P.
315	FXY	REAL			273	I	INTEGER		
0	INFY	REAL	ARRAY	F.P.	274	J	INTEGER		
316	K	INTEGER			276	K1	INTEGER		
277	K2	INTEGER			2.5	4	INTEGER		
	N	INTEGER		F.P.	2.3	MUMX	REAL		
2..	MUMY	REAL				DF1	REAL		F.P.
0	DF2	REAL		F.P.	0	S	REAL	ARRAY	F.P.
311	X	REAL			3.2	XP	REAL		
3.4	X1	REAL			256	X1P	REAL		
2.4	X1FP	REAL			3.5	X2	REAL		
25.	X2F	REAL			2.5	X2PP	REAL		
31.	X3	REAL			262	X3P	REAL		
27.	X3FF	REAL			3..	Y	REAL		
3.3	YP	REAL			3..	Y1	REAL		
257	Y1F	REAL			2.5	Y1PP	REAL		
3.7	Y2	REAL			2.1	Y2P	REAL		
267	Y2FP	REAL			311	Y3	REAL		

```
      SUBROUTINE LHF(DATA3,LH)
      DIMENSION DATA3(8,8),LH(64,4)
      REAL LH
C*** COMPUTATION OF DELH/DELX
C     COMPUTATION OF EDGES
      DO 10 I=1,8
      I1= (I-1)*8 + 1
      I2= (I-1)*8 + 8
      LH(I1,1) = (DATA3(I,2)-DATA3(I,1))/2.
      LH(I1,3)=LH(I1,1)
      LH(I2,1)= (DATA3(I,8)-DATA3(I,7))/2.
      LH(I2,3)=LH(I2,1)
   10 CONTINUE
C     COMPUTATION OF MIDDLE
      DO 11 I=1,8
      DO 11 J=2,7
      I3= (I-1)*8 + J
      LH(I3,1) = (DATA3(I,J+1)-DATA3(I,J-1))/4.
      LH(I3,3)=LH(I3,1)
   11 CONTINUE
C*** COMPUTATION OF DELH/DELY
C     COMPUTATION OF EDGES
      DO 20 I=1,8
      I4=I+56
      LH(I,2)= (DATA3(2,I)-DATA3(1,I))/2.
      LH(I,4)=LH(I,2)
      LH(I4,2)= (DATA3(8,I)-DATA3(7,I))/2.
      LH(I4,4)=LH(I4,2)
   20 CONTINUE
C     COMPUTATION OF MIDDLE
      K=8
      DO 21 I=2,7
      DO 21 J=1,8
      K=K+1
      LH(K,2)= (DATA3(I+1,J)-DATA3(I-1,J))/4.
      LH(K,4)=LH(K,2)
   21 CONTINUE
      PRINT*,"LINEARIZED H FUNCTION"
   22 FORMAT(5X,4(F12.4,2X))
      DO 23 I=1,64
      PRINT22,LH(I,1),LH(I,2),LH(I,3),LH(I,4)
   23 CONTINUE
      RETURN
      END
```

SYMBOLIC REFERENCE MAP (R=1)

ENTRY POINTS
3 LHF

VARIABLES SN TYPE RELOCATION
DATA3 REAL ARRAY F.P. 145 I INTEGER
I1 INTEGER INTEGER

```
      SUBROUTINE NOISE(DATA1,R,SN,V,PHIN)
      DIMENSION DATA1(24,24),R(64,64),V(64),W(64),PHIN(64,64),WNO(64)
      DIMENSION WKA(64)
      COMPLEX DATA1
      N=64
      IF(SN.EQ.0.0)GO TO 10
      IF(SN.EQ.1.0)GO TO 11
      IF(SN.EQ.2.0)GO TO 12
      IF(SN.EQ.3.0)GO TO 12
C*** ADDITION OF SPATIAL NOISE TO INPUT MATRIX
C     GENERATION OF GUASSIAN VECTOR W
11    CALL VEC(N,W)
C*** COMPUTATION OF V=R*W
      L=64
      M=64
      N=1
      IA=64
      IB=64
      IC=64
      CALL VMULFF(R,W,L,M,N,IA,IB,V,IC,IER)
      GO TO 20
C*** ADDITION OF TEMPORAL OR TEMPORAL WITH SPATIAL NOISE TO INPUT
C     GENERATION OF GUASSIAN VECTOR
12    CALL VEC(N,W)
C*** COMPUTATION OF WNO=R*W
      L=64
      M=64
      N=1
      IA=64
      IB=64
      IC=64
      CALL VMULFF(R,W,L,M,N,IA,IB,WNO,IC,IER)
C***  COMPUTATION OF NOISE VECTOR V (V=PHIN*V)
      CALL VMULFF(PHIN,V,L,M,N,IA,IB,WKA,IC,IER)
      DO 31 I=1,64
      V(I)=WKA(I)
31    CONTINUE
C****ADDITION OF V TO WNO
      DO 21 I=1,64
      V(I)=WKA(I) + WNO(I)
21    CONTINUE
C****ADDITION OF NOISE TO INPUT ARRAY
20    K=0
      DO 22 I=1,8
      DO 22 J=1,8
      K=K + 1
      DATA1(I+8,J+8) = DATA1(I+8,J+8) + V(K)
22    CONTINUE
10    RETURN
      END
```

Best Available Copy

SYMBOLIC REFERENCE MAP (R=1)

```
      SUBROUTINE ACC(DF1)
      IDK=DF1
      RDK=DF1-IDK
      PRINT*,"IDK=",IDK
      PRINT*,"RDK=",RDK
      R=ABS(RDK)
      IF(R.LT. .)RDK= .
      DF1=RDK + IDK
      RETURN
      END
```

SYMBOLIC REFERENCE MAP (R=1)

```
ENTRY POINTS
    3  ACC

VARIABLES     SN  TYPE          RELOCATION
   35  R          REAL                         34  IDK        INTEGER
    0  DF1        REAL               F.P.      35  RDK        REAL

FILE NAMES          MODE
      OUTPUT        FMT

INLINE FUNCTIONS    TYPE     ARGS
      ABS           REAL        1  INTRIN

STATISTICS
  PROGRAM LENGTH                    38      35
  ...  CM USED
```

```
SUBROUTINE ANAY        74/74   OPT=1  PMDMP                 FTN 4.8+510          1/29/80  21.02.27        PAGE    1

            SUBROUTINE ANAY(XM,CENX,CENY,K3,SAV)
            DIMENSION SAV(1[illegible],[illegible]),XM(6)
            SAV(K3,1)=XM(1) + XM(3) +2[illegible].
            SAV(K3,2) =CENY
            SAV(K3,3)=XM(2) + XM(4) +2[illegible].
            SAV(K3,4)=CENY
            SAV(K3,1)= SAV(K3,1) - CENX
            SAV(K3,3)= SAV(K3,3) - CENY
            RETURN
            END
```

```
SYMBOLIC REFERENCE MAP (R=1)

ENTRY POINTS
    3  ANAY

VARIABLES      SN   TYPE                 RELOCATION
    0  CENX         REAL                    F.P.              0  CENY      REAL               F.P.
    0  K3           INTEGER                 F.P.              0  SAV       REAL     ARRAY     F.P.
    0  XM           REAL          ARRAY     F.P.

STATISTICS
  PROGRAM LENGTH                  23B      19
      [illegible] CM USED
```

Best Available Copy

```
ROUTINE CHSKI        74/74     OPT=1   PMDMP                    FTN 4.8+313        1/29/73   21.42.27        PAGE    1

               SUBROUTINE CHSKI(A,B,N)
               DIMENSION A(N,N),B(N,N)
               DO 1  I=1,N
               DO 1  J=1,N
               IF(J.GT.I) GO TO 1
               IF(J.EQ.I) GO TO 2
         C*** LOWER TRIANGULAR ELEMENT
               IF(J.NE.1) GO TO 4
               B(I,J)= (A(I,J))/B(J,J)
               GO TO 1
         4     RES= 0.
               M1=J-1
               DO 11 K=1,M1
               RES= B(I,K)*B(J,K) + RES
         11    CONTINUE
               B(I,J) = (A(I,J) - RES)/B(J,J)
               GO TO 1
         C*** DIAGONAL ELEMENTS
         2     IF(I.NE.1) GO TO 3
               B(I,J) =SQRT(A(I,J))
               GO TO 1
         3     RES= 0.
               M2=I-1
               DO 12 K=1,M2
               RES =RES + B(I,K)**2
         12    CONTINUE
               B(I,J)=SQRT(A(I,I)-RES)
               GO TO 1
         C*** UPPER TRIANGULAR ELEMENTS WILL BE SET TO ZERO
         1     B(I,J)=0.
         1     CONTINUE
               RETURN
               END


      SYMBOLIC REFERENCE MAP (R=1)

   ENTRY POINTS
      3  CHSKI

   VARIABLES    SN   TYPE          RELOCATION
      0  B           REAL         ARRAY    F.P.        0  B         REAL       ARRAY    F.P.
    116  I           INTEGER                         117  J         INTEGER
    122  K           INTEGER                         121  M1        INTEGER
    123  M2          INTEGER                           0  N         INTEGER             F.P.
    124  RES         REAL

   EXTERNALS          TYPE    ARGS
      SQRT            REAL      1 LIBRARY

   STATEMENT LABELS
```

Best Available Copy 78

```
                 SUBROUTINE MWRITE(A,N,M,IDIM)
                 DIMENSION A(IDIM,IDIM)
                 PRINT*," "
                 DO 350 IK=1,N
                 PRINT 340,(A(IK,JK),JK=1,M)
          340    FORMAT(1X,10(F7.4,1X))
          350    CONTINUE
                 PRINT*," "
                 RETURN
                 END
```

```
      SYMBOLIC REFERENCE MAP (R=1)

 ENTRY POINTS
     3  MWRITE

 VARIABLES     SN  TYPE          RELOCATION
     0  A          REAL        ARRAY   F.P.            0  IDIM     INTEGER      F.P.
    55  IK         INTEGER                            55  JK       INTEGER
     0  M          INTEGER              F.P.           0  N        INTEGER      F.P.

 FILE NAMES           MODE
      OUTPUT          MIXED

 STATEMENT LABELS
    47  340       FMT                        0  350

 LOOPS  LABEL    INDEX      FROM-TO     LENGTH      PROPERTIES
    13  350      IK          4  7        2?3                     EXT REFS   NOT INNER
    16           JK          5  5        1?3                     EXT REFS

 STATISTICS
   PROGRAM LENGTH                 ?7B       55
        ?? ? ?B CM USED
```

```
      SUBROUTINE PROP(XP,PP,PHI,QD,XM,PM)
      DIMENSION XP(4),PP(4,4),PHI(4,4),QD(4,4),XM(4),PM(4,4)
     1 ,WKA(4,4),WKB(4,4)
C***PROPAGATION OF STATE VECTOR XM=PHI*XP
      L=4
      M=4
      N=1
      IA=4
      IB=4
      IC=4
      CALL VMULFF(PHI,XP,L,M,N,IA,IB,XM,IC,IER)
      PRINT *,"X MINUS VECTOR "
   10 FORMAT(1X,/,1X,"X0= ",F10.7,2X,"Y0= ",F10.7,2X,
     1 "XA= ",F10.7,2X,"YA= ",2X,F10.7,////)
      PRINT 10,XM(1),XM(2),XM(3),XM(4)
C***PROPAGATION OF COVARIANCE MATRIX
C     PM= PHI*PP*PHIT + QD
C     SINCE PHI IS A DIAGONAL MATRIX, PHIT=PHI
      N=4
      CALL VMULFF(PHI,PP,L,M,N,IA,IB,WKA,IC,IER)
      CALL VMULFF(WKA,PHI,L,M,N,IA,IB,WKB,IC,IER)
      DO 2 I=1,4
      DO 2 J=1,4
      PM(I,J)= WKB(I,J) + QD(I,J)
    2 CONTINUE
      PRINT *, "COVARIANCE MINUS MATRIX"
      PRINT*," "
   11 FORMAT(1X,4(F10.7,2X))
      DO 25 I=1,4
      PRINT 11,(PM(I,J),J=1,4)
   25 CONTINUE
      PRINT*," "
      PRINT*," "
      PRINT*," "
      RETURN
      END
```

SYMBOLIC REFERENCE MAP (R=1)

ENTRY POINTS
3 PROP

```
VARIABLES    SN  TYPE            RELOCATION
  227 I          INTEGER                          223 IA       INTEGER
  224 IB         INTEGER                          225 IC       INTEGER
  226 IER        INTEGER                          230 J        INTEGER
  22. L          INTEGER                          221 N        INTEGER
  222 M          INTEGER                            . PHI      REAL      ARRAY  F.P.
    . PM         REAL       ARRAY   F.P.            . PP       REAL      ARRAY  F.P.
    . QD         REAL       ARRAY   F.P.          231 WKA      REAL      ARRAY
  2251 WKB       REAL       ARRAY                   . XM       REAL      ARRAY  F.P.
    . XP         REAL       ARRAY   F.P.
```

Best Available Copy

```
      SUBROUTINE SHIFT(DATA1,DF1,DF2)
      DIMENSION DATA1(2,,24),WK1(24,24)
      COMPLEX DATA1,WK1,XX,YY,YY1,YY2,XX1,XX2
C*** REARRANGE ARRAY OUT OF FOURT FORMAT
C     SWITCH QUADRANTS 2 AND 4
      DO 1  I=1,12
      DO 1  J=1,12
      WK1(I+12,J+12)=DATA1(I,J)
      WK1(I,J)=DATA1(I+12,J+12)
 1    CONTINUE
C     SWITCH QUADRANTS 1 AND 3
      DO 11 I=1,12
      DO 11 J=13,24
      WK1(I+12,J-12)=DATA1(I,J)
      WK1(I,J)=DATA1(I+12,J-12)
 11   CONTINUE
C*** IMPLEMENTATION OF PHASE SHIFT (DC TERM AT (13,13))
      DEN=2.*12.
      PI=3.14159265
      DO 10 I=1,11
      I1=13-I
      I2=13+I
      J1=13-I
      J2=13+I
C     LEFT AND RIGHT SIDES OF HARMONIC BLOCK
      FY1=J1-13
      FY2=J2-13
      FYY1=2.*PI*FY1*DF2/DEM
      FYY2=2.*PI*FY2*DF2/DEM
      YY1=CMPLX( . ,FYY1)
      YY2=CMPLX( . ,FYY2)
      DO 13 I1=I1,I2
      FX=I1-13
      FXX=2.*PI*FX*DF1/DEM
      XX=CMPLX( . ,FXX)
      WK1(I1,J1)=WK1(I1,J1)*CEXP(XX+YY1)
      WK1(I1,J2)=WK1(I1,J2)*CEXP(XX+YY2)
 13   CONTINUE
C     TOP AND BOTTOM OF HARMONIC BLOCK
      FX1=I1-13
      FX2=I2-13
      FXX1=2.*PI*FX1*DF1/DEM
      FXX2=2.*PI*FX2*DF1/DEM
      XX1=CMPLX( . ,FXX1)
      XX2=CMPLX( . ,FXX2)
      JA1=J1+1
      JA2=J2-1
      DO 14 JA=JA1,JA2
      FY=JA-13
      FYY=2.*PI*FY*DF2/DEM
      YY=CMPLX( . ,FYY)
      WK1(I1,JA)=WK1(I1,JA)*CEXP(XX1+YY)
      WK1(I2,JA)=WK1(I2,JA)*CEXP(XX2+YY)
 14   CONTINUE
 12   CONTINUE
C*** SPECIAL CASE TO HANDLE LEFT SIDE OF 12TH HARMONIC BLOCK
      FY=-12
```

```
      FYY=2.*PI*FY*DF2/DIM
      YY=CMPLX(0.,FYY)
      DO 17 I=1,24
      FX=I-13
      FXX=2.*PI*FX*DF1/DIM
      XX=CMPLX(0.,FXX)
      WKA(I,1)=WKA(I,1)*CEXP(XX+YY)
17    CONTINUE
C***SPECIAL CASE TO HANDLE TOP OF 12TH HARMONIC BLOCK
      FX=-12
      FXX=2.*PI*FX*DF1/DIM
      XX=CMPLX(0.,FXX)
      DO 18 J=2,24
      FY=J-13
      FYY=2.*PI*FY*DF2/DIM
      YY=CMPLX(0.,FYY)
      WKA(1,J)=WKA(1,J)*CEXP(XX+YY)
18    CONTINUE
C***REARRANGE ARRAY BACK INTO FOURT FORMAT
C     SWITCH QUADRANTS 2 AND 4
      DO 1  I=1,12
      DO 1  J=1,12
      DATA1(I+12,J+12)=WKA(I,J)
      DATA1(I,J)=WKA(I+12,J+12)
1     CONTINUE
C     SWITCH QUADRANTS 1 AND 3
      DO 15 I=1,12
      DO 15 J=13,24
      DATA1(I+12,J-12)=WKA(I,J)
      DATA1(I,J)=WKA(I+12,J-12)
15    CONTINUE
      RETURN
      END
```

SYMBOLIC REFERENCE MAP (R=1)

ENTRY POINTS
3 SHIFT

VARIABLES		SN	TYPE	RELOCATION				
0	DATA1		COMPLEX	ARRAY	F.P.	352	DIM	REAL
355	FX		REAL			356	FXX	REAL
371	FXX1		REAL			372	FXX2	REAL
357	FX1		REAL			370	FX2	REAL
376	FY		REAL			377	FYY	REAL
362	FYY1		REAL			363	FYY2	REAL
360	FY1		REAL			361	FY2	REAL
354	I		INTEGER			364	IA	INTEGER
354	I1		INTEGER			355	I2	INTEGER
351	J		INTEGER			375	JA	INTEGER
373	JA1		INTEGER			374	JA2	INTEGER
366	J1		INTEGER			367	J2	INTEGER
	DF1		REAL		F.P.		DF2	REAL

Best Available Copy

```
      SUBROUTINE SPIN(C,SIGMAR,R,NN)
      DIMENSION R(64,64),C(5),NN(64,64)
C
C     SET UP SPATIAL NOISE CORRELATION COEFFICIENT MATRIX
C
      N=64
      M=8
      DO 30 I=1,N
      DO 30 J=1,N
      R(I,J)=0.
   30 CONTINUE
      DO 35 I=1,N
      R(I,I)=1.
      IF (I.GE.64) GO TO 35
      R(I,I+1)=C(1)
      IF (I.GE.63) GO TO 35
      R(I,I+2)=C(2)
      IF(I.GE.[illegible]) GO TO 35
      R(I,I+[illegible])=C([illegible])
      IF(I.GE.[illegible]) GO TO 35
      R(I,I+[illegible])=C(2)
      IF(I.GE.57) GO TO 35
      R(I,I+[illegible])=C(1)
      IF(I.GE.[illegible]) GO TO 35
      R(I,I+[illegible])=C(2)
      IF(I.GE.[illegible]) GO TO 35
      R(I,I+[illegible])=C(4)
      IF(I.GE.[illegible]) GO TO 35
      R(I,I+[illegible])=C([illegible])
      IF(I.GE.[illegible]) GO TO 35
      R(I,I+15)=C(4)
      IF(I.GE.[illegible]) GO TO 35
      R(I,I+16)=C(3)
      IF(I.GE.48) GO TO 35
      R(I,I+17)=C(4)
      IF(I.GE.47) GO TO 35
      R(I,I+15)=C(5)
   35 CONTINUE
      DO 37 I=1,8
      R([illegible]-7,[illegible]I)=0.0
      R([illegible]I-7,M*I-1)=0.0
      R(M*I-5,8*I)=0.0
      IF (I.GE.8) GO TO 37
      R([illegible]I,[illegible]I+1)=0.
      R([illegible]I,M*I+2)=0.
      R([illegible]I-1,M*I+1)=0.0
      R(8*I-[illegible],M*I+7)=0.0
      R(8*I-7,8*I+8)=0.0
      R(8*I-[illegible],8*I+9)=0.0
      IF (I.GE.7) GO TO 37
      R([illegible]I,M*I+9)=0.0
      R(M*I,M*I+1[illegible])=0.
      R(M*I-1,M*I+9)=0.0
      IF (I.GE.[illegible]) GO TO 37
      R(M*I,M*I+17)=0.
      R(M*I,M*I+18)=0.
      R(8*I-1,8*I+17)=0.0
```

```
   37    CONTINUE
         DO 31 I=1,N
         L=I+1
         DO 31 J=L,N
         IF(L.GT.N) GO TO 75
         R(J,I)=R(I,J)
   31    CONTINUE
         DO 33 I=1,N
         DO 33 J=1,N
         RN(I,J)=SIGMAS R(I,J)
   33    CONTINUE
         PRINT 44
   44    FORMAT(///2X,"N X N SPATIAL CORRELATION MATRIX:"/)
         CALL MWRITE (RN,...,54)
C
C          COMPUTE THE CHOLESKY SQUARE ROOT OF RN
C
   75    CALL CHESKI(RN,N,J)
         PRINT 43
   43    FORMAT(///2X,"THE CHOLESKY SQUARE ROOT OF RN:"/)
         CALL MWRITE (R,N,...,54)
         RETURN
         END
```

SYMBOLIC REFERENCE MAP (R=1)

ENTRY POINTS
   3 SPIN

```
VARIABLES       SN  TYPE            RELOCATION
    ..  C           REAL     ARRAY     F.P.        241  I           INTEGER
   242  J           INTEGER                        243  L           INTEGER
   244  M           INTEGER                        237  N           INTEGER
     .  R           REAL     ARRAY     F.P.          .  RN          REAL     ARRAY   F.P.
     .  SIGMAS      REAL               F.P.
```

```
FILE NAMES        MODE
   OUTPUT         FMT
```

```
EXTERNALS         TYPE   ARGS
   CHSKI                   -                   MWRITE
```

```
STATEMENT LABELS
    0  31                         66  35                        113  37
  173  35                          0  33                        216  44      FMT
  227  48      FMT
```

```
LOOPS  LABEL  INDEX   FROM-TO   LENGTH   PROPERTIES
   11   3.     I        6 11      137                 NOT INNER
   16   3.     J        9 11       23     INSTACK
   32   36     I       12 36      363       OPT
   ..   37     I       34 35      173       OPT
  117   38     I       59 64      213                 NOT INNER
```

Best Available Copy 84

```
      SUBROUTINE UPD(Z,HLM,LH,XM,PM,RINV,XP,PP)
      DIMENSION Z(4),HLM(4),LH(4,4),LHT(4,4)
     1 ,XM(4),PM(4,4),XP(4),PP(4,4),PINV(4,4),HTH(4,4),
     2 WKA(4),WKS(4),K(4,4),RES(4)
      REAL HLM,LH,LHT,K
C**  *COMPUTATION OF COVARIANCE PLUS MATRIX USING INVERSE COVARIANCE FORM
      DO 10 I=1,4
      DO 10 J=1,4
      LHT(J,I)=LH(I,J)
 10   CONTINUE
      L=4
      M=4
      N=4
      IA=4
      IB=4
      IC=4
      CALL VMULFF(LHT,LH,L,M,N,IA,IB,HTH,IC,IER)
      IDGT=0
      CALL LINV2F(PM,N,IA,PINV,IDGT,WKA,IER)
      PRINT*,"IDGT= ",IDGT
      DO 11 I=1,4
      DO 11 J=1,4
      PINV(I,J)= PINV(I,J) + RINV*HTH(I,J)
 11   CONTINUE
      IDGT=0
      CALL LINV2F(PINV,N,IA,PP,IDGT,WKA,IER)
      PRINT*,"IDGT= ",IDGT
      PRINT*,"COVARIANCE MATRIX"
      PRINT*," "
 12   FORMAT(1X,4(E11.3,2X))
      DO 13 I=1,4
      PRINT12, (PP(I,J),J=1,4)
 13   CONTINUE
C**  *COMPUTATION OF KALMAN FILTER GAIN
      L=4
      M=4
      N=4
      IA=4
      IB=4
      IC=4
      CALL VMULFF(PP,LHT,L,M,N,IA,IB,K,IC,IER)
      DO 14 I=1,4
      DO 14 J=1,4
      K(I,J)= RINV*K(I,J)
 14   CONTINUE
      PRINT*,"KALMAN FILTER GAIN"
      DO 15 J=1,4
      PRINT*,(K(I,J),I=1,4)
 15   CONTINUE
C**  *COMPUTATION OF STATE MEASUREMENT UPDATE
      DO 16 I=1,4
      RES(I)=Z(I)-HLM(I)
 16   CONTINUE
      PRINT*,"RESIDUES"
      DO 17 J=1,4
      PRINT*,RES(J)
```

```
                 M=??
                 N=1
 6               I??=
                 CALL VMULFF(P,RES,L,M,N,IA,IB,WK3,IC,IER)
                 DO 15 I=1,4
                 XP(I)=XP(I) + WK3(I)
         15      CONTINUE
 65              PRINT*,"STATE VECTOR"
         17      FORMAT(1X,/,"XP= ",F11.7,"YD= ",F11.7,
               1 "XA= ",F11.7,"YA= ",F11.7,///)
                 PRINT17,XP(1),XP(2),XP(3),XP(4)
                 RETURN
 7               END
```

```
SYMBOLIC REFERENCE MAP (R=1)

ENTRY POINTS
    3  UPD

VARIABLES     SN  TYPE        RELOCATION
  1?1  HTP        REAL        ARRAY                    3?5  I      INTEGER
  353  IA         INTEGER                              35?  IB     INTEGER
  355  IC         INTEGER                              357  IDGT   INTEGER
  356  IER        INTEGER                              3?7  J      INTEGER
  1?72 K          REAL        ARRAY                    3??  L      INTEGER
    ?  LH         REAL        ARRAY     F.P.           3??  LHT    REAL     ARRAY
  351  M          INTEGER                              352  N      INTEGER
    ?  NLM        REAL        ARRAY     F.P.           7?   PINV   REAL     ARRAY
       PM         REAL        ARRAY     F.P.             ?  PP     REAL     ARRAY   F.P.
  1?72 RES        REAL        ARRAY                      ?  PANV   REAL             F.P.
  1??  WKA        REAL        ARRAY                   1?65  WK3    REAL     ARRAY
    ?  YM         REAL        ARRAY     F.P.             ?  XP     REAL     ARRAY   F.P.
    ?  Z          REAL        ARRAY     F.P.

FILE NAMES        MODE
     OUTPUT       MIXED

EXTERNALS         TYPE   ARGS
     LINV2F               7                        VMULFF            10

STATEMENT LABELS
    ?  1?                          ?  11                    3?7  12     FMT
    ?  ?3                          ?  14                      ?  15
    6  16                        337  17     FMT              ?  13
    ?  19

LOOPS  LABEL  INDEX   FROM-TO   LENGTH   PROPERTIES
  ?7   1?     I        ? 1?      14?                    NOT INNER
  15   1?     J        5 1?      3?      INSTACK
  41   11     I       21 24      1??                    NOT INNER
  66   11     J       22 24      ?9      INSTACK
  7?   13     I       31 33      1?3     ?  EXT REFS   NOT INNER
  7?   ??     J       32 3?      ???     ?  EXT REFS
```

```
 1                      SUBROUTINE VEC(N,M)
                        DIMENSION M(N)
                        DO 15 J=1,N
                        TOTAL = .
                        DO 5 I=1,12
 5                      TOTAL = TOTAL + RANF(DUM)
                 5      CONTINUE
                        M(J)=TOTAL - 6.0
                15      CONTINUE
 10                     RETURN
                        END
```

```
        SYMBOLIC REFERENCE MAP (R=1)

ENTRY POINTS
   3  VEC

VARIABLES    SN  TYPE          RELOCATION
   31  DUM       REAL       UNDEF                     30  I         INTEGER
   2*  J         INTEGER                               0  N         INTEGER               F.P.
   27  TOTAL     REAL                                  0  M         REAL         ARRAY    F.P.

INLINE FUNCTIONS    TYPE    ARGS
      RANF          REAL      1  INTRIN

STATEMENT LABELS
      5                                 0  15

LOOPS  LABEL     INDEX      FROM-TO    LENGTH    PROPERTIES
   7  15         J           3 5        1.8                 NOT INNER
  13  5          I           5 7        .3        INSTACK

STATISTICS
  PROGRAM LENGTH                 31B       25
     1.....B 31 USED
```

```
      SUBROUTINE FOURT(DATA,NN,NDIM,ISIGN,IFORM,WORK)
C     FOR INFORMATION CONTACT DR. MARK HALLER 43E/ADDS/36286
C
C     THE COOLEY-TUKEY FAST FOURIER TRANSFORM IN USASI BASIC FORTRAN
C
C     TRANSFORM(K1,K2,...) = SUM(DATA(J1,J2,...)*EXP(ISIGN*2*PI*SQRT(-1)
C     *((J1-1)*(K1-1)/NN(1)+(J2-1)*(K2-1)/NN(2)+...))), SUMMED FOR ALL
C     J1, K1 BETWEEN 1 AND NN(1), J2, K2 BETWEEN 1 AND NN(2), ETC.
C     THERE IS NO LIMIT TO THE NUMBER OF SUBSCRIPTS.  DATA IS A
C     MULTIDIMENSIONAL COMPLEX ARRAY WHOSE REAL AND IMAGINARY
C     PARTS ARE ADJACENT IN STORAGE, SUCH AS FORTRAN IV PLACES THEM.
C     IF ALL IMAGINARY PARTS ARE ZERO (DATA ARE DISGUISED REAL), SET
C     IFORM TO ZERO TO CUT THE RUNNING TIME BY UP TO FORTY PERCENT.
C     OTHERWISE, IFORM = +1.  THE LENGTHS OF ALL DIMENSIONS ARE
C     STORED IN ARRAY NN, OF LENGTH NDIM.  THEY MAY BE ANY POSITIVE
C     INTEGERS, THO THE PROGRAM RUNS FASTER ON COMPOSITE INTEGERS, AND
C     ESPECIALLY FAST ON NUMBERS RICH IN FACTORS OF TWO.  ISIGN IS +1
C     OR -1.  IF A -1 TRANSFORM IS FOLLOWED BY A +1 ONE (OR A +1
C     BY A -1) THE ORIGINAL DATA REAPPEAR, MULTIPLIED BY NTOT (=NN(1)*
C     NN(2)*...).  TRANSFORM VALUES ARE ALWAYS COMPLEX, AND ARE RETURNED
C     IN ARRAY DATA, REPLACING THE INPUT.  IN ADDITION, IF ALL
C     DIMENSIONS ARE NOT POWERS OF TWO, ARRAY WORK MUST BE SUPPLIED,
C     COMPLEX OF LENGTH EQUAL TO THE LARGEST NON 2**K DIMENSION.
C     OTHERWISE, REPLACE WORK BY ZERO IN THE CALLING SEQUENCE.
C     NORMAL FORTRAN DATA ORDERING IS EXPECTED, FIRST SUBSCRIPT VARYING
C     FASTEST.  ALL SUBSCRIPTS BEGIN AT ONE.
C
C     RUNNING TIME IS MUCH SHORTER THAN THE NAIVE NTOT**2, BEING
C     GIVEN BY THE FOLLOWING FORMULA.  DECOMPOSE NTOT INTO
C     2**K2 * 3**K3 * 5**K5 * ....  LET SUM2 = 2*K2, SUMF = 3*K3 + 5*K5
C     + ... AND NF = K3 + K5 + ....  THE TIME TAKEN BY A MULTI-
C     DIMENSIONAL TRANSFORM ON THESE NTOT DATA IS T = T0 + NTOT*(T1+
C     T2*SUM2+T3*SUMF+T4*NF).  ON THE CDC 3300 (FLOATING POINT ADD TIME
C     OF SIX MICROSECONDS), T = 3000 + NTOT*(500+43*SUM2+68*SUMF+
C     320*NF) MICROSECONDS ON COMPLEX DATA.  IN ADDITION, THE
C     ACCURACY IS GREATLY IMPROVED, AS THE RMS RELATIVE ERROR IS
C     BOUNDED BY 3*2**(-B)*SUM(FACTOR(J)**1.5), WHERE B IS THE NUMBER
C     OF BITS IN THE FLOATING POINT FRACTION AND FACTOR(J) ARE THE
C     PRIME FACTORS OF NTOT.
C
C     PROGRAM BY NORMAN BRENNER FROM THE BASIC PROGRAM BY CHARLES
C     RADER.  RALPH ALTER SUGGESTED THE IDEA FOR THE DIGIT REVERSAL.
C     MIT LINCOLN LABORATORY, AUGUST 1967.  THIS IS THE FASTEST AND MOST
C     VERSATILE VERSION OF THE FFT KNOWN TO THE AUTHOR.  SHORTER PRO-
C     GRAMS FOUR1 AND FOUR2 RESTRICT DIMENSION LENGTHS TO POWERS OF TWO.
C     SEE-- IEEE AUDIO TRANSACTIONS (JUNE 1967), SPECIAL ISSUE ON FFT.
C
C     THE DISCRETE FOURIER TRANSFORM PLACES THREE RESTRICTIONS UPON THE
C     DATA.
C     1.  THE NUMBER OF INPUT DATA AND THE NUMBER OF TRANSFORM VALUES
C     MUST BE THE SAME.
C     2.  BOTH THE INPUT DATA AND THE TRANSFORM VALUES MUST REPRESENT
C     EQUISPACED POINTS IN THEIR RESPECTIVE DOMAINS OF TIME AND
C     FREQUENCY.  CALLING THESE SPACINGS DELTAT AND DELTAF, IT MUST BE
C     TRUE THAT DELTAF=2*PI/(NN(I)*DELTAT).  OF COURSE, DELTAT NEED NOT
C     BE THE SAME FOR EVERY DIMENSION.
```

```
C     3.  CONCEPTUALLY AT LEAST, THE INPUT DATA AND THE TRANSFORM OUTPUT
C     REPRESENT SINGLE CYCLES OF PERIODIC FUNCTIONS.
C
C     EXAMPLE 1.  THREE-DIMENSIONAL FORWARD FOURIER TRANSFORM OF A
C     COMPLEX ARRAY DIMENSIONED 32 BY 25 BY 13 IN FORTRAN IV.
C     DIMENSION DATA(32,25,13),WORK(50),NN(3)
C     COMPLEX DATA
C     DATA NN/32,25,13/
C     DO 1 I=1,32
C     DO 1 J=1,25
C     DO 1 K=1,13
C   1 DATA(I,J,K)=COMPLEX VALUE
C     CALL FOURT(DATA,NN,3,-1,1,WORK)
C
C     EXAMPLE 2.  ONE-DIMENSIONAL FORWARD TRANSFORM OF A REAL ARRAY OF
C     LENGTH 64 IN FORTRAN II.
C     DIMENSION DATA(2,64)
C     DO 2 I=1,64
C     DATA(1,I)=REAL PART
C   2 DATA(2,I)=0.
C     CALL FOURT(DATA,64,1,-1,0,0)
C
      DIMENSION DATA(1),NN(1),IFACT(32),WORK(1)
C     ... INITIALIZATION
      WR=0.
      WI=0.
      WSTPR=0.
      WSTPI=0.
      TWOPI=6.28318530
      IF(NDIM-1)920,1,1
    1 NTOT=2
      DO 2 IDIM=1,NDIM
      IF(NN(IDIM))920,920,2
    2 NTOT=NTOT*NN(IDIM)
C
C     MAIN LOOP FOR EACH DIMENSION
C
      NP1=2
      DO 910 IDIM=1,NDIM
      N=NN(IDIM)
      NP2=NP1*N
      IF(N-1)920,900,5
C
C     FACTOR N
C
    5 M=N
      NTWO=NP1
      IF=1
      IDIV=2
   10 IQUOT=M/IDIV
      IREM=M-IDIV*IQUOT
      IF(IQUOT-IDIV)50,11,11
   11 IF(IREM)20,12,20
   12 NTWO=NTWO+NTWO
      M=IQUOT
      GO TO 10
```

```
   30 IQUOT=M/IDIV
      IREM=M-IDIV*IQUOT
      IF(IQUOT-IDIV)50,31,31
   31 IF(IREM)40,32,40
   32 IFACT(IF)=IDIV
      IF=IF+1
      M=IQUOT
      GO TO 30
   40 IDIV=IDIV+2
      GO TO 30
   50 IF(IREM)60,51,60
   51 NTWO=NTWO+NTWO
      GO TO 70
   60 IFACT(IF)=M
C
C     SEPARATE FOUR CASES--
C        1. COMPLEX TRANSFORM OR REAL TRANSFORM FOR THE 4TH, 5TH,ETC.
C           DIMENSIONS.
C        2. REAL TRANSFORM FOR THE 2ND OR 3RD DIMENSION.  METHOD--
C           TRANSFORM HALF THE DATA, SUPPLYING THE OTHER HALF BY CON-
C           JUGATE SYMMETRY.
C        3. REAL TRANSFORM FOR THE 1ST DIMENSION, N ODD.  METHOD--
C           TRANSFORM HALF THE DATA AT EACH STAGE, SUPPLYING THE OTHER
C           HALF BY CONJUGATE SYMMETRY.
C        4. REAL TRANSFORM FOR THE 1ST DIMENSION, N EVEN.  METHOD--
C           TRANSFORM A COMPLEX ARRAY OF LENGTH N/2 WHOSE REAL PARTS
C           ARE THE EVEN NUMBERED REAL VALUES AND WHOSE IMAGINARY PARTS
C           ARE THE ODD NUMBERED REAL VALUES.  SEPARATE AND SUPPLY
C           THE SECOND HALF BY CONJUGATE SYMMETRY.
C
   70 NON2=NP1*(NP2/NTWO)
      ICASE=1
      IF(IDIM-4)71,90,90
   71 IF(ICPLX)72,72,90
   72 ICASE=2
      IF(IDIM-1)73,73,90
   73 ICASE=3
      IF(NTWO-NP1)90,90,74
   74 ICASE=4
      NTWO=NTWO/2
      N=N/2
      NP2=NP2/2
      NTOT=NTOT/2
      I=3
      DO 80 J=2,NTOT
      DATA(J)=DATA(I)
   80 I=I+2
   90 I1RNG=NP1
      IF(ICASE-2)100,95,100
   95 I1RNG=NP0*(1+NPREV/2)
C
C     SHUFFLE ON THE FACTORS OF TWO IN N.  AS THE SHUFFLING
C     CAN BE DONE BY SIMPLE INTERCHANGE, NO WORKING ARRAY IS NEEDED
C
  100 IF(NTWO-NP1)600,600,110
  110 NP2HF=NP2/2
      J=1
```

```
      DO 130 I2=I1,NP2,NON2
      IF(J-I2)120,130,130
120   I1MAX=I2+NON2-2
      DO 125 I1=I2,I1MAX,2
      DO 125 I3=I1,NTOT,NP2
      J3=J+I3-I2
      TEMPR=DATA(I3)
      TEMPI=DATA(I3+1)
      DATA(I3)=DATA(J3)
      DATA(I3+1)=DATA(J3+1)
      DATA(J3)=TEMPR
125   DATA(J3+1)=TEMPI
130   M=NP2HF
140   IF(J-M)150,150,145
145   J=J-M
      M=M/2
      IF(M-NON2)150,140,140
150   J=J+M
C
C     MAIN LOOP FOR FACTORS OF TWO.  PERFORM FOURIER TRANSFORMS OF
C     LENGTH FOUR, WITH ONE OF LENGTH TWO IF NEEDED.  THE TWIDDLE FACTOR
C     W=EXP(ISIGN*2*PI*SQRT(-1)*M/(4*MMAX)).  CHECK FOR W=ISIGN*SQRT(-1)
C     AND REPEAT FOR W=ISIGN*SQRT(-1)*CONJUGATE(W).
C
      NON2T=NON2+NON2
      IPAR=NP2HF/NON2
310   IF(IPAR-2)350,330,320
320   IPAR=IPAR/4
      GO TO 310
330   DO 340 I1=1,I1RNG,2
      DO 340 J3=I1,NON2,NP1
      DO 340 K1=J3,NTOT,NON2T
      K2=K1+NON2
      TEMPR=DATA(K2)
      TEMPI=DATA(K2+1)
      DATA(K2)=DATA(K1)-TEMPR
      DATA(K2+1)=DATA(K1+1)-TEMPI
      DATA(K1)=DATA(K1)+TEMPR
340   DATA(K1+1)=DATA(K1+1)+TEMPI
350   MMAX=NON2
360   IF(MMAX-NP2HF)370,600,600
370   LMAX=MAX0(NON2T,MMAX/2)
      IF(MMAX-NON2)405,405,380
380   THETA=-TWOPI*FLOAT(NON2)/FLOAT(4*MMAX)
      IF(ISIGN)400,390,390
390   THETA=-THETA
400   WR=COS(THETA)
      WI=SIN(THETA)
      WSTPR=-2.*WI*WI
      WSTPI=2.*WR*WI
405   DO 570 L=NON2,LMAX,NON2T
      M=L
      IF(MMAX-NON2)420,420,410
410   W2R=WR*WR-WI*WI
      W2I=2.*WR*WI
      W3R=W2R*WR-W2I*WI
      W3I=W2R*WI+W2I*WR
```

```
 420  DO 570 I1=1,I1RNG,2
      DO 570 J3=I1,NON2,NP1
      KMIN=J3+IFACT*
      IF(MMAX-NP(M3)) 430,430,440
 430  KMIN=J3
 440  KDIF=IPA*MMAX
 450  KSTEP=4*KDIF
      DO 520 K1=KMIN,NTOT,KSTEP
      K2=K1+KDIF
      K3=K2+KDIF
      K4=K3+KDIF
      IF(MMAX-NP(M2))-L3,460,480
 460  U1R=DATA(K1)+DATA(K2)
      U1I=DATA(K1+1)+DATA(K2+1)
      U2R=DATA(K3)+DATA(K4)
      U2I=DATA(K3+1)+DATA(K4+1)
      U3R=DATA(K1)-DATA(K2)
      U3I=DATA(K1+1)-DATA(K2+1)
      IF(ISIGN) 470,475,475
 470  U4R=DATA(K3+1)-DATA(K4+1)
      U4I=DATA(K4)-DATA(K3)
      GO TO 510
 475  U4R=DATA(K4+1)-DATA(K3+1)
      U4I=DATA(K3)-DATA(K4)
      GO TO 510
 480  T2R=W2R*DATA(K2)-W2I*DATA(K2+1)
      T2I=W2R*DATA(K2+1)+W2I*DATA(K2)
      T3R=WR*DATA(K3)-WI*DATA(K3+1)
      T3I=WR*DATA(K3+1)+WI*DATA(K3)
      T4R=W3R*DATA(K4)-W3I*DATA(K4+1)
      T4I=W3R*DATA(K4+1)+W3I*DATA(K4)
      U1R=DATA(K1)+T2R
      U1I=DATA(K1+1)+T2I
      U2R=T3R+T4R
      U2I=T3I+T4I
      U3R=DATA(K1)-T2R
      U3I=DATA(K1+1)-T2I
      IF(ISIGN) 490,500,500
 490  U4R=T3I-T4I
      U4I=T4R-T3R
      GO TO 510
 500  U4R=T4I-T3I
      U4I=T3R-T4R
 510  DATA(K1)=U1R+U2R
      DATA(K1+1)=U1I+U2I
      DATA(K2)=U3R+U4R
      DATA(K2+1)=U3I+U4I
      DATA(K3)=U1R-U2R
      DATA(K3+1)=U1I-U2I
      DATA(K4)=U3R-U4R
 520  DATA(K4+1)=U3I-U4I
      KMIN=4*(KMIN-J3)+J3
      KDIF=KSTEP

      KDIF=KSTEP
      IF(KDIF-NP1) 530,530,53
 570  CONTINUE
```

```
      MMAX=MMAX-M
      IF(ISIGN) ...
320   TEMPR=WR
      WR=-WI
      WI=-TEMPR
      GO TO ...
330   TEMPR=WR
      WR=WI
      WI=TEMPR
335   IF(M-LMAX)337,...
337   TEMPR=WR
      WR=WR*WSTPR-WI*WSTPI+WR
340   WI=WI*WSTPR+TEMPR*WSTPI+WI
      IPAR=3-IPAR
      MMAX=MMAX+MMAX
      GO TO 300
C
C     MAIN LOOP FOR FACTORS NOT EQUAL TO TWO.  APPLY THE TWIDDLE FACTOR
C     W=EXP(ISIGN*2*PI*SQRT(-1)*(J2-1)*(J1-J2)/(NP2*IFP1)), THEN
C     PERFORM A FOURIER TRANSFORM OF LENGTH IFACT(IF), MAKING USE OF
C     CONJUGATE SYMMETRIES.
C
500   IF(NTWO-NP2)605,700,700
505   IFP1=NON2
      IF=1
      NP1HF=NP1/2
510   IFP2=IFP1/IFACT(IF)
      J1RNG=NP2
      IF(ICASE-3)512,511,512
511   J1RNG=(NP2+IFP1)/2
      J2STP=NP2/IFACT(IF)
      J1RG2=(J2STP+IFP2)/2
512   J2MIN=1+IFP2
      IF(IFP1-NP2)515,540,540
515   DO 535 J2=J2MIN,IFP1,IFP2
      THETA=-TWOPI*FLOAT(J2-1)/FLOAT(NP2)
      IF(ISIGN)525,520,520
520   THETA=-THETA
525   SINTH=SIN(THETA/2.)
      WSTPR=-2.*SINTH*SINTH
      WSTPI=SIN(THETA)
      WR=WSTPR+1.
      WI=WSTPI
      J1MIN=J2+IFP1
      DO 535 J1=J1MIN,J1RNG,IFP1
      I1MAX=J1+I1RNG-2
      DO 530 I1=J1,I1MAX,2
      DO 530 I3=I1,NTOT,NP2
      J3MAX=I3+IFP2-NP1
      DO 530 J3=I3,J3MAX,NP1
      TEMPR=DATA(J3)
      DATA(J3)=DATA(J3)*WR-DATA(J3+1)*WI
530   DATA(J3+1)=TEMPR*WI+DATA(J3+1)*WR
      TEMPR=WR
      WR=WR*WSTPR-WI*WSTPI+WR
535   WI=TEMPR*WSTPI+WI*WSTPR+WI
      ...IFACT(IF)
```

Best Available Copy

```
      IF(ISIGN)5..,5..,5..
  5.. THETA=-THETA
  5.. SINTH=SIN(THETA/2.)
      WSTPR=-2.*SINTH*SINTH
      WSTPI=SIN(THETA)
      NSTEP=NP1/IFACT(IF)
      MSING=KSTEP*(IFACT(IF)/2)+1
      DO 5.. I1=1,I1RNG,2
      DO 5.. I3=I1,NTOT,NP2
      DO 5.. KMIN=1,KRANG,KSTEP
      J1MAX=I3+J1..NG-IFP1
      DO 5.. J1=I3,J1MAX,IFP1
      J3MAX=J1+IFP2-NP1
      DO 5.. J3=J1,J3MAX,NP1
      J2MAX=J3+IFP1-IFP2
      K=KMIN+(J2-J1+(J1-I3)/IFACT(IF))/NP1HF
      IF(KMIN-1)55.,55.,555
  55. SUMR=0.
      SUMI=0.
      DO 5.. J2=J3,J2MAX,IFP2
      SUMR=SUMR+DATA(J2)
  55. SUMI=SUMI+DATA(J2+1)
      WORK(K)=SUMR
      WORK(K+1)=SUMI
      GO TO 5..
  55. KCONJ=K+2*(N-KMIN+1)
      J2=J2MAX
      SUMR=DATA(J2)
      SUMI=DATA(J2+1)
      OLDSR=0.
      OLDSI=0.
      J2=J2-IFP2
  57. TEMPR=SUMR
      TEMPI=SUMI
      SUMR=TWOWR*SUMR-OLDSR+DATA(J2)
      SUMI=TWOWR*SUMI-OLDSI+DATA(J2+1)
      OLDSR=TEMPR
      OLDSI=TEMPI
      J2=J2-IFP2
      IF(J2-J3)5..,5..,57.
  575 TEMPR=WR*SUMR-OLDSR+DATA(J2)
      TEMPI=WI*SUMI
      WORK(K)=TEMPR-TEMPI
      WORK(KCONJ)=TEMPR+TEMPI
      TEMPR=WR*SUMI-OLDSI+DATA(J2+1)
      TEMPI=WI*SUMR
      WORK(K+1)=TEMPR+TEMPI
      WORK(KCONJ+1)=TEMPR-TEMPI
  5.. CONTINUE
      IF(K-IM-1)..5,5..,5..
  5.. WR=WSTPR+1.
      WI=WSTPI
      GO TO 5..
  5.. TWOWR=WR
      WR=WR*WSTPR-WI*WSTPI+WR
      WI=TEMPR*WSTPI+WI*WSTPR+WI
  5.. TWOWR=WR+WR
```

```
      IF(ISTATE-3)692,691,692
691   IF(IFP1-NP2)692,692,692
692   K=1
      I2MAX=I3+NP2-NP1
      DO 693 I2=I3,I2MAX,NP1
      DATA(I2)=WORK(K)
      DATA(I2+1)=WORK(K+1)
693   K=K+2
      GO TO 698
C
C     COMPLETE A REAL TRANSFORM IN THE 1ST DIMENSION, N ODD, BY CON-
C     JUGATE SYMMETRIES AT EACH STAGE.
C
695   J3MAX=I3+IFP2-NP1
      DO 697 J3=I3,J3MAX,NP1
      J2MAX=J3+NP2-J2STP
      DO 697 J2=J3,J2MAX,J2STP
      J1MAX=J2+J1RNG2-IFP2
      J1CNJ=J3+J2MAX+J2STP-J2
      DO 697 J1=J2,J1MAX,IFP2
      K=1+J1-I3
      DATA(J1)=WORK(K)
      DATA(J1+1)=WORK(K+1)
      IF(J1-J2)696,697,696
696   DATA(J1CNJ)=WORK(K)
      DATA(J1CNJ+1)=-WORK(K+1)
697   J1CNJ=J1CNJ-IFP2
698   CONTINUE
      IF=IF+1
      IFP1=IFP2
      IF(IFP1-NP1)700,700,601
C
C     COMPLETE A REAL TRANSFORM IN THE 1ST DIMENSION, N EVEN, BY CON-
C     JUGATE SYMMETRIES.
C
700   GO TO (900,800,900,701),ICASE
701   NHALF=N
      N=N+N
      THETA=-TWOPI/FLOAT(N)
      IF(ISIGN)702,703,703
702   THETA=-THETA
703   SINTH=SIN(THETA/2.)
      WSTPR=-2.*SINTH*SINTH
      WSTPI=SIN(THETA)
      WR=WSTPR+1.
      WI=WSTPI
      IMIN=3
      JMIN=2*NHALF-1
      GO TO 725
71    J=JMIN
      DO 720 I=IMIN,NTOT,NP2
      SUMR=(DATA(I)+DATA(J))/2.
      SUMI=(DATA(I+1)+DATA(J+1))/2.
      DIFR=(DATA(I)-DATA(J))/2.
      DIFI=(DATA(I+1)-DATA(J+1))/2.
      TEMPR=WR*SUMI+WI*DIFR
      TEMPI=WI*SUMI-WR*DIFR
```

Best Available Copy

```
      DATA(I)=SUMR+TEMPR
      DATA(I+1)=DIFI+TEMPI
      DATA(J)=SUMR-TEMPR
      DATA(J+1)=-DIFI+TEMPI
 720  J=J+NP2
      IMIN=IMIN+2
      IMAX=IMAX-2
      TEMPR=WR
      WR=WR*WSTPR-WI*WSTPI+WR
      WI=TEMPR*WSTPI+WI*WSTPR+WI
 725  IF(IMIN-IMAX)710,730,740
 730  IF(ISIGN)731,740,740
 731  DO 735 I=IMIN,NTOT,NP2
 735  DATA(I+1)=-DATA(I+1)
 740  NP2=NP2+NP2
      NTOT=NTOT+NTOT
      J=NTOT+1
      IMAX=NTOT/2+1
 745  IMIN=IMAX-2*NHALF
      I=IMIN
      GO TO 755
 750  DATA(J)=DATA(I)
      DATA(J+1)=-DATA(I+1)
 755  I=I+2
      J=J-2
      IF(I-IMAX)750,760,760
 760  DATA(J)=DATA(IMIN)-DATA(IMIN+1)
      DATA(J+1)=0.
      IF(I-J)770,780,780
 765  DATA(J)=DATA(I)
      DATA(J+1)=DATA(I+1)
 770  I=I-2
      J=J-2
      IF(I-IMIN)775,775,765
 775  DATA(J)=DATA(IMIN)+DATA(IMIN+1)
      DATA(J+1)=0.
      IMAX=IMIN
      GO TO 745
 780  DATA(1)=DATA(1)+DATA(2)
      DATA(2)=0.
      GO TO 900
C
C     COMPLETE A REAL TRANSFORM FOR THE 2ND OR 3RD DIMENSION BY
C     CONJUGATE SYMMETRIES.
C
 800  IF(I1RNG-NP1)805,900,900
 805  DO 860 I3=1,NTOT,NP2
      I2MAX=I3+NP2-NP1
      DO 860 I2=I3,I2MAX,NP1
      IMIN=I2+I1RNG
      IMAX=I2+NP1-2
      JMAX=2*I3+NP1-IMIN
      IF(I2-I3)820,820,810
 810  JMAX=JMAX+NP2
 820  IF(IDIM-2)850,850,830
 830  J=JMAX+NP0
      DO 840 I=IMIN,IMAX,2
```

```
                    DATA(I)=DATA(J)                                          FFTT
      51            DATA(I+1)=-DATA(J+1)                                     FFTT
               30   J=J-2                                                    FFTT
               35   J=JMAX                                                   FFTT
                    DO 52 I=IMIN,IMAX,NP                                     FFTT
                    DATA(I)=DATA(J)                                          FFTT
      52            DATA(I+1)=-DATA(J+1)                                     FFTT
               30   J=J-NP                                                   FFTT
              C                                                              FFTT
              C     END OF LOOP ON EACH DIMENSION                            FFTT
              C                                                              FFTT
      524      310  NP0=NP1                                                  FFTT
                    NP1=NP2                                                  FFTT
               31   NPREV=N                                                  FFTT
               32   RETURN                                                   FFTT
                    END                                                      FFTT
```

```
CARD NR. SEVERITY  DETAILS     DIAGNOSIS OF PROBLEM

    49       I      DATA       ARRAY REFERENCE OUTSIDE DIMENSION BOUNDS.
    49       I      DATA       ARRAY REFERENCE OUTSIDE DIMENSION BOUNDS.
```



```
     SYMBOLIC REFERENCE MAP (R=1)

 ENTRY POINTS
      3   FOURT

 VARIABLES      SN   TYPE              RELOCATION
      1  DATA        REAL        ARRAY       F.P.          1527  DIF1      REAL
   1526  DIF         REAL                                  1417  I         INTEGER
   14    ICASE       INTEGER                               14 3  IDIM      INTEGER
   1412  IDIV        INTEGER                               1411  IF        INTEGER
   1372  IFACT       INTEGER     ARRAY                           IFORM     INTEGER         F.P.
   1473  IFP1        INTEGER                               1473  IFP2      INTEGER
   153   IMAX        INTEGER                               1524  IMIN      INTEGER
   1435  IPAR        INTEGER                               1413  IQUOT     INTEGER
   1414  IREM        INTEGER                                     ISIGN     INTEGER         F.P.
   1 2   I1          INTEGER                               1 2   I1MAX     INTEGER
   1 21  I1RNG       INTEGER                               142   I2        INTEGER
   1521  I2MAX       INTEGER                               143   I3        INTEGER
   142   J           INTEGER                               1531  JMAX      INTEGER
   1525  JMIN        INTEGER                               15 5  J1        INTEGER
   1522  J1CNJ       INTEGER                               1 1   J1MAX     INTEGER
   15    J1MIN       INTEGER                               1     J1RG2     INTEGER
   147   J1RNG       INTEGER                               15 2  J2        INTEGER
   1511  J2MAX       INTEGER                               1  1  J2MIN     INTEGER
   14    J2STP       INTEGER                               1431  J3        INTEGER
   15    J3MAX       INTEGER                               1512  K         INTEGER
   1515  KCONJ       INTEGER                               1451  KDIF      INTEGER
   14    KMIN        INTEGER                               15 7  KRANG     INTEGER
```

VITA

James Singletery Jr. was born on November 4, 1955 in Lackawanna, New York. He graduated from Lackawanna Senior High School in June 1973 and entered the United States Air Force Academy that same month. In June of 1977, he graduated from the Air Force Academy with a Bachelor of Science degree in Electrical Engineering and a commission in the United States Air Force. From July 1977 to June 1979, he served as a telemetry developmental engineer with the 4950th Test Wing at Wright-Patterson Air Force Base, Ohio. In June 1979, First Lieutenant James Singletery Jr. was assigned to the Air Force Institute of Technology to pursue a Master's of Science Degree in Electrical Engineering (Electro-Optics). As a result of graduate work done at night during his assignment with the Test Wing, he earned a Master's of Science Degree in Management Science in April 1980 from the University of Dayton. Lt. Singletery is also a member of the Eta Kappa Nu Society.

Permanent address: 11 Clark St.
Lackawanna, New York 14218

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER AFIT/GEO/EE/80D-12	2. GOVT ACCESSION NO. AD-A100 870	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) ADAPTIVE LASER POINTING AND TRACKING PROBLEM		5. TYPE OF REPORT & PERIOD COVERED MS Thesis
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) James Singletery Jr., 1st Lt.		8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS Air Force Institute of Technology (AFIT-EN) Wright-Patterson AFB, Ohio 45433		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Air Force Weapons Laboratory/ALO Kirtland AFB, NM 87117		12. REPORT DATE December, 1980
		13. NUMBER OF PAGES 108
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		15. SECURITY CLASS. (of this report) Unclassified
		15a. DECLASSIFICATION DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

Approved for public release; IAW AFR 190-17

Fredric C. Lynch

Fredric C. Lynch, Major, USAF
Director of Public Affairs, AFIT

16 JUN 1981

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Kalman Filter
Pattern Recognition
Fast Fourier Transform

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

Although a number of the major objectives that were established at the outset of this project were not met, a number of milestones were realized. The digital implementation of a negating phase shift that operates perfectly under ideal conditions was a major accomplishment. The establishment of a zero level of 10^{-8} was also significant. The incorporation of the exponential smoothing technique to minimize the effect of measurement noise was important since it uncovered a possible connection between the size of the target image and its

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

20. ABSTRACT

performance throughout the pattern recognition process. However, the major obstacle that surfaced during the execution of this project was a filter divergence problem. It has been proposed that this problem can be solved by implementing the Fourier transform derivative property instead of the forward-backward difference method to compute the spatial derivative of the non-linear h function.